NASA/CP—2018-219785

Toolbox for the Modeling and Analysis of Thermodynamic System (T-MATS) Users' Workshop Presentations

Jonathan S. Litt, Compiler
Glenn Research Center, Cleveland, Ohio

April 2018

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program plays a key part in helping NASA maintain this important role.

The NASA STI Program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI Program provides access to the NASA Technical Report Server—Registered (NTRS Reg) and NASA Technical Report Server—Public (NTRS) thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counter-part of peer-reviewed formal professional papers, but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., "quick-release" reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at http://www.sti.nasa.gov

- E-mail your question to help@sti.nasa.gov

- Fax your question to the NASA STI Information Desk at 757-864-6500

- Telephone the NASA STI Information Desk at 757-864-9658

- Write to:
  NASA STI Program
  Mail Stop 148
  NASA Langley Research Center
  Hampton, VA 23681-2199

NASA/CP—2018-219785

Toolbox for the Modeling and Analysis of Thermodynamic System (T-MATS) Users' Workshop Presentations

Jonathan S. Litt, Compiler
Glenn Research Center, Cleveland, Ohio

Proceedings of a conference held at the Ohio Aerospace Institute
sponsored by NASA Glenn Research Center
Cleveland, Ohio
August 21, 2017

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

April 2018

Acknowledgments

The organizers would like to thank the NASA Transformational Tools and Technologies (TTT) project, under the Transformative Aeronautics Concepts Program (TACP) for funding the on-going development of T-MATS.

This work was sponsored by the Transformative Aeronautics Concepts Program.

Level of Review: This material has been technically reviewed by technical management.

Available from

NASA STI Program
Mail Stop 148
NASA Langley Research Center
Hampton, VA 23681-2199

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161
703-605-6000

This report is available in electronic form at http://www.sti.nasa.gov/ and http://ntrs.nasa.gov/

Contents

T-MATS Users' Workshop

Abstract

NASA Glenn Research Center hosted a Users' Workshop on the Toolbox for the Modeling and Analysis of Thermodynamic Systems (T-MATS) on August 21, 2017. The objective of this workshop was to update the user community on the latest features of T-MATS, and to provide a forum to present work performed using T-MATS. Presentations highlighted creative applications and the development of new features and libraries, and emphasized the flexibility and simulation power of T-MATS.

Introduction

The Toolbox for the Modeling and Analysis of Thermodynamic Systems (T-MATS) was developed at NASA Glenn Research Center to facilitate the rapid generation of turbomachinery simulations in a standardized environment. T-MATS is an Open Source graphical thermodynamic simulation package built in MATLAB/Simulink (The MathWorks, Inc). It combines generic thermodynamic and controls modeling capability with a numerical iterative solver to create a framework for the creation of complex simulations. A feature of the package is the turbomachinery block set. This set of Simulink blocks gives a developer the tools required to create virtually any steady-state or dynamic turbomachinery simulation, e.g., a gas turbine simulation. In systems where the control or other related components are modeled in MATLAB/Simulink, the T-MATS developer has the ability to create the complete system in a single tool.

T-MATS was originally released in 2014, and is making an impact within NASA on a variety of aeronautics projects. As T-MATS approaches 5000 external downloads, conference and journal papers are beginning to appear documenting its use. Additionally, the developers of T-MATS at NASA Glenn are aware of several unpublished proprietary applications. Based on this success and the desire to encourage interaction between users to further community development of the Open Source software, a T-MATS Users' Workshop was planned. The 2017 T-MATS Users' Workshop provided a forum for developers to describe new features being incorporated into T-MATS, as well as for researchers to present new applications of interest to the user community. The Workshop consisted of two sessions. The first covered the in-house development activities; the second covered applications, both in-house and out-of-house, as well as a new functionality developed by an outside entity.

The following sections contain the presentations from the Workshop. Some of this work has appeared in the literature previously, other work is new and was shown for the first time at the Workshop. A list of references is included after the presentations to give the reader some background in T-MATS' capabilities, and applications that utilize T-MATS for some aspect of the work reported. It is the sincere hope of the Workshop organizers that this information will enable the reader to recognize the power, flexibility, and ease of use provided by T-MATS, and to consider it for future applications.

T-MATS is available for download at: https://github.com/nasa/T-MATS/releases

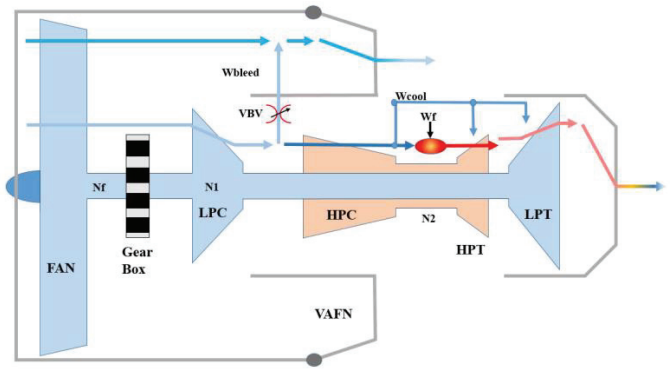
NASA/CP—2018-219785

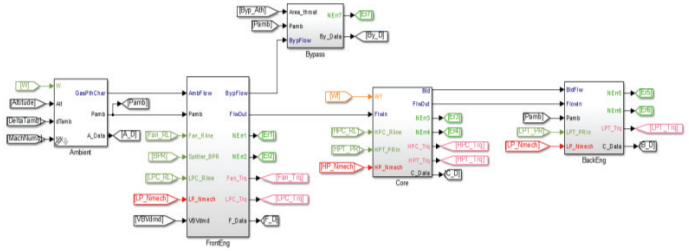
Toolbox for the Modeling and Analysis of Thermodynamic Systems (T-MATS) Users' Workshop

In-House Development Activities

Welcome/Overview

- The Toolbox for the Modeling and Analysis of Thermodynamic Systems is an open-source, graphical simulation package developed at NASA GRC
- It is primarily used to model gas turbines, but has been used to model other thermodynamic systems as well
- It is built using MATLAB/Simulink and C code, and is a plug-in to Simulink





Welcome/Overview

- T-MATS was released in 2014
- First T-MATS Workshop held in April 2015
- T-MATS' use is growing, with over 4500 downloads to date
- At least two journal articles describing research that involved the use of T-MATS appeared recently
- Used by NASA, industry, and academia, with internal NASA use spread across Aeronautics Programs

NASA/CP—2018-219785

T-MATS Development

- T-MATS is an Open Source Software package created at NASA Glenn
- Internal T-MATS development is supported by projects
- Internal T-MATS development does not generally occur for its own sake
- However, Open Source software encourages collaborative development, and user-defined blocks can be posted on our github site
- A desired Workshop Outcome is to define a prioritized list of desired improvements
- This list can be used to advocate for additional development

NASA/CP—2018-219785

Expectations

- Attendees will learn about T-MATS and its current features
- Presenters will describe various T-MATS applications, current and planned
- Model developers will talk about their experience with T-MATS:
  - What features are good and why?
  - Can they be improved?
  - What is missing?
- The group will discuss new feature development options. Can we reach a consensus?
- The group will prioritize these potential new features

NASA/CP—2018-219785

Agenda T-MATS Users' Workshop

1:00 PM	Welcome	10	Jonathan S. Litt
1:10 PM	T-MATS Overview	80	T-MATS Overview and Recent Capability Additions/NPSS T-MATS Relationship/NASA Applications--Jeffryes W. Chapman T-MATS Volumetric Blocks--Aidan W. Rinehart General use Geared Turbofan Simulation in T-MATS and demo--Jeffryes W. Chapman
2:30 PM	Demo	20	Jeffryes W. Chapman
2:50 PM	Break	10	
3:00 PM	Outside Presentations	80	T-MATS analysis of a rotating detonation engine--Guillermo Paniagua, Purdue University Modeling an Aircraft Propulsion Subsystem for Developing Coordinating Controllers in a More Electric Aircraft Using T-MATS--William Dunham, University of Michigan Applications of T-MATS to Hardware-in-the-Loop Simulation Modeling--George Thomas, N&R Engineering Embedded controller code generation from T-MATS blocks--Jason Whitfield, MathWorks, Inc.
4:20 PM	Discussion on Future Capabilities/ Needs	40	All
5:00 PM	Adjourn		

NASA/CP—2018-219785

Toolbox for the Modeling and Analysis of Thermodynamic Systems (T-MATS)

2nd T-MATS Workshop
Ohio Aerospace Institute (OAI)
Cleveland, OH
August 21, 2017

Team

- Jeffryes W. Chapman

  Vantage Partners, LLC. Cleveland, OH 44142

- Thomas M. Lavelle

  NASA Glenn Research Center, Cleveland, OH 44135

- Jonathan S. Litt

  NASA Glenn Research Center, Cleveland, OH 44135

- Aidan W. Rinehart

  Vantage Partners, LLC. Cleveland, OH 44142

NASA/CP—2018-219785

Outline

- T-MATS Overview
  - Description
  - General Use
- Features
  - Types of Blocks
  - Advanced Capabilities
- New Features and Updates
- Project Role and Status
- Summary

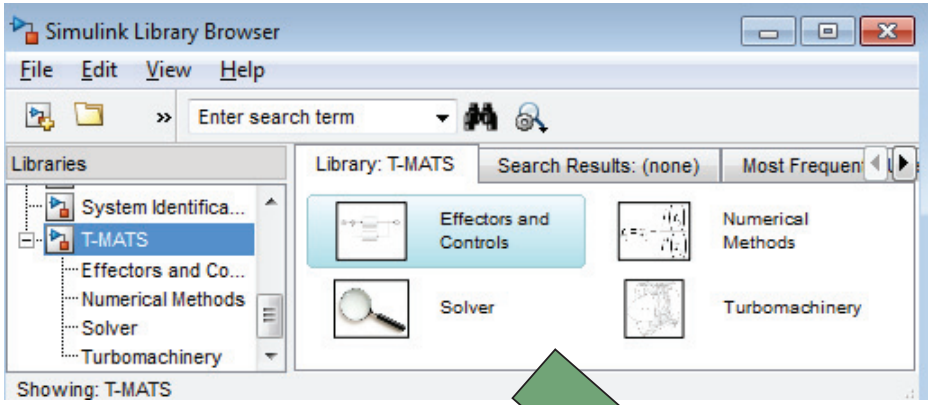
NASA/CP—2018-219785

T-MATS Description

- Toolbox for the Modeling and Analysis of Thermodynamic systems, T-MATS
  - Modular thermodynamic modeling framework
  - Designed for easy creation of custom Component Level Models (CLM)
  - Built in MATLAB®/Simulink®
- Package highlights
  - General thermodynamic simulation design framework
  - Variable input system solvers
  - Advanced turbo-machinery block sets
  - Control system block sets
- Development being led by NASA Glenn Research Center
  - Non-proprietary, free of export restrictions, and open source
    - Open collaboration environment

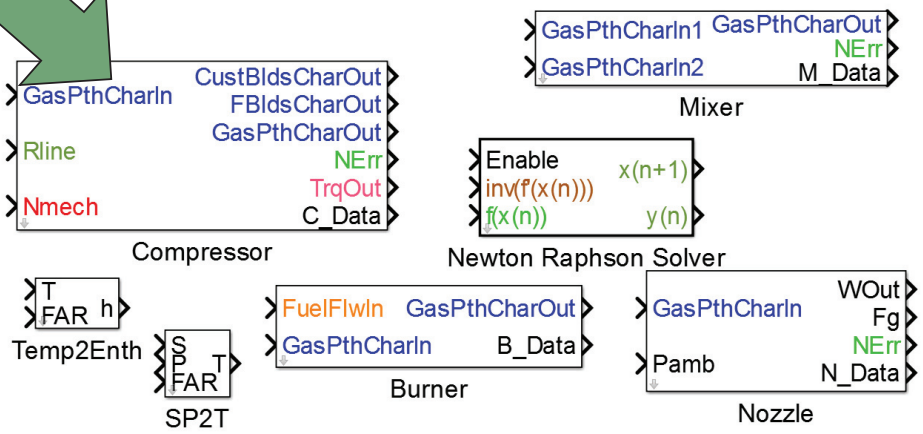
T-MATS Framework

- Plug-in for the industry-standard MATLAB/Simulink platform
  - additional blocks in the Simulink Library Browser:



Added Simulink Thermodynamic modeling and numerical solving functionality

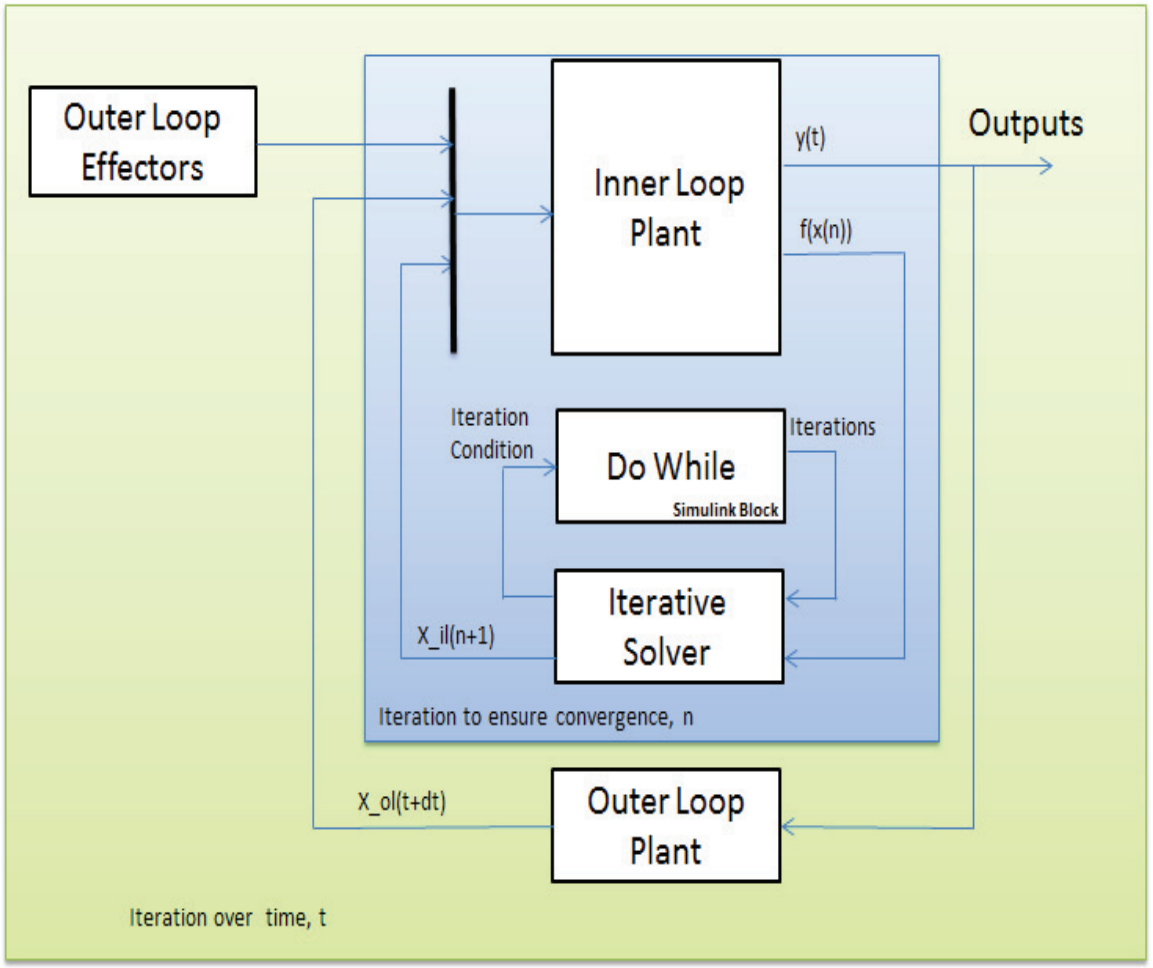
Faster and easier model creation



T-MATS Framework

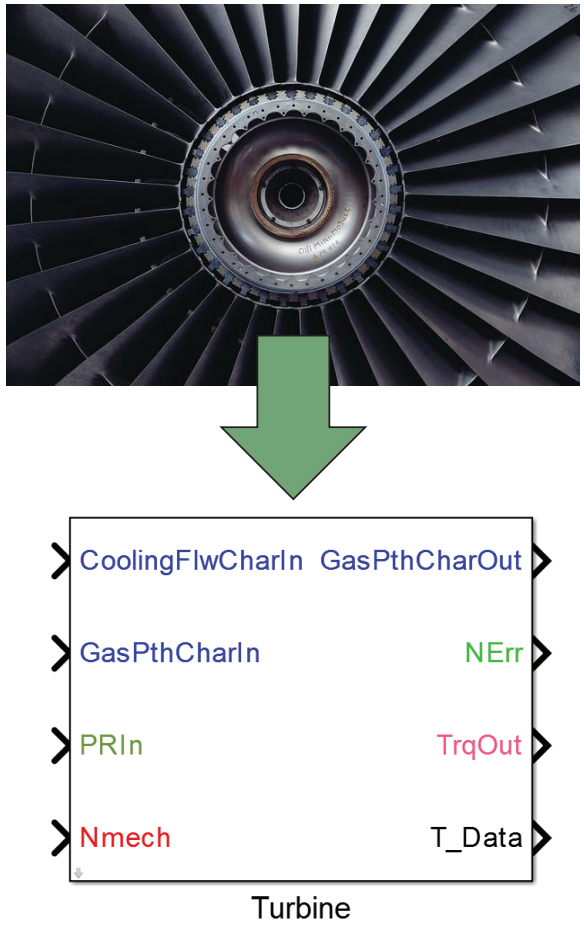
Dynamic Simulation Example:

- Multi-loop structure
  - The “outer” loop (green) iterates in the time domain
    - Not required for steady-state models
  - The “inner” loop (blue) solves for plant convergence during each time step



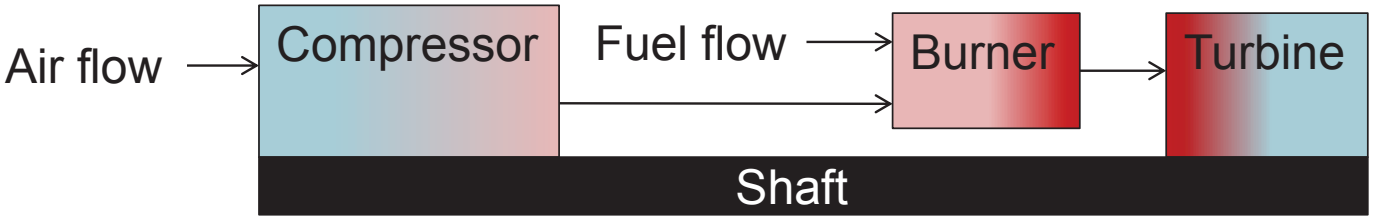
Blocks: Turbo-machinery

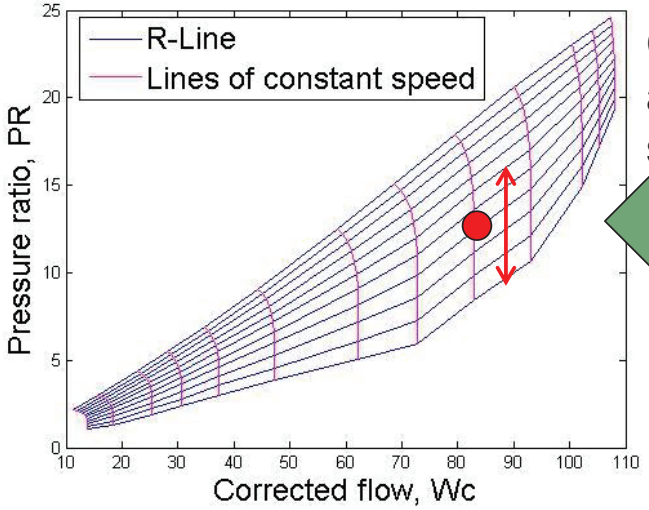
- T-MATS contains component blocks necessary for creation of turbo-machinery systems
  - Modeling theory based on common industry practices
    - 0-D flow components, $W_{in} = W_{out}$
    - Energy balance modeling approach
    - Compressor models utilize R-line compressor maps
    - Turbine models utilize Pressure Ratio turbine maps
  - Blocks types: compressor, turbine, nozzle, flow splitter, and valves among others.
    - Color Coding for easy setup
  - Built with S-functions, utilizing compiled C code/ MEX functions



Blocks: Numerical Solver

- T-MATS contains libraries of solvers based on the Newton Raphson method to ensure system convergence.
- Why is an external solver necessary?
  - In gas turbines, air flow through the engine is dependent on system architecture and a solver is required to achieve a balance the flow.

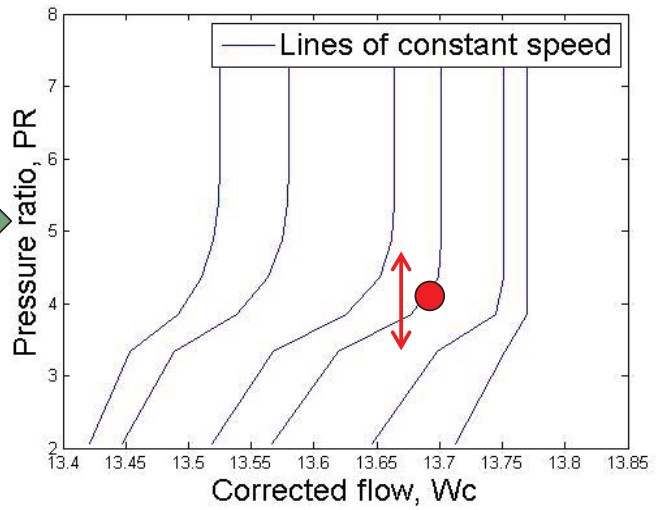




Components must agree on W for the system

Convergence

Effectors:
Shaft speed
Pressure
Temperature



Blocks: Controls

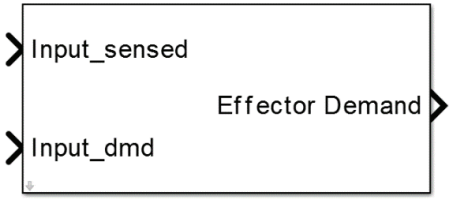
- T-MATS contains component blocks designed for fast control system creation
  - General Design
    - Sensors:

1st order Sensor

    - Actuators:

1st order Actuator

    - PI controllers:

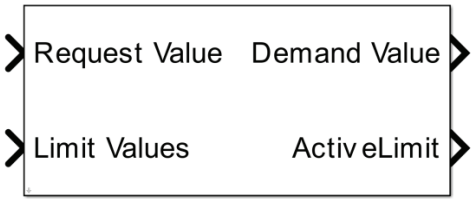


Simple PI controller

  - Engine Design
    - PI Regulator Controller:

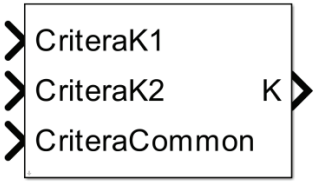
Limit Controller (PI)

    - Limit selection logic:



Limit Selection

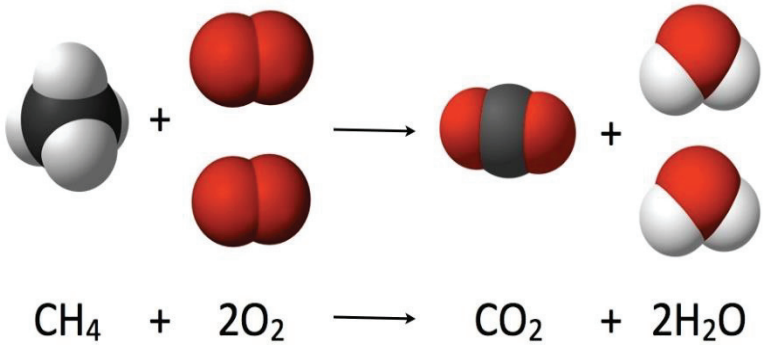
    - Standardized table lookups:



GainSchedulerD2D

Advanced Capabilities

- Integration with Cantera
  - Cantera models chemical kinetics, thermodynamics, and/or transport properties.
  - It is C++ based code with interfaces for python, MATLAB, C, and Fortran 90 (Code-based and open source)
  - Enables modeling of fuel cells, engines using alternative fuels, etc.
  - Integration with T-MATS enables Cantera's capabilities to be utilized in a graphical plug and play modeling environment



https://en.wikipedia.org/wiki/File:Combustion_reaction_of_methane.jpg

Combustion reaction of methane

Simplification

T-MATS custom class based scripts and blocks simplify Cantera and allow easy creation of complex systems.

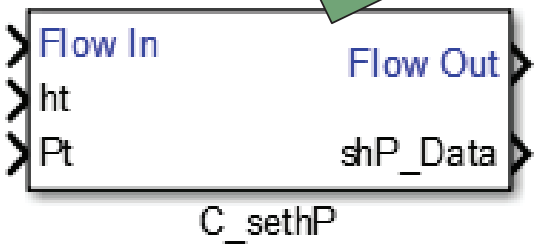
```
while  abs (lasterr)>.000000001 && count < 50
 set(fs,'Y',obj.CompVal_Can);
 set(fs, 'T', Ttg*5./9.,'P',Ptg*6894.75729 );
 equilibrate(fs, 'TP');
 htg = enthalpy_mass( fs )*.0004302099943161011;
 root = htg-htOut;
 sec_out = TMATSC.FlowDef.iterSecant( root, Ttg, last,
   lasterr, .1 );
 next = sec_out(1);
 last = sec_out(2);
 lasterr = sec_out(3);
 Ttg = next;
end
```

Cantera Code

```
TMATSC.set_hP(FlowObj,ht,Pt)
```

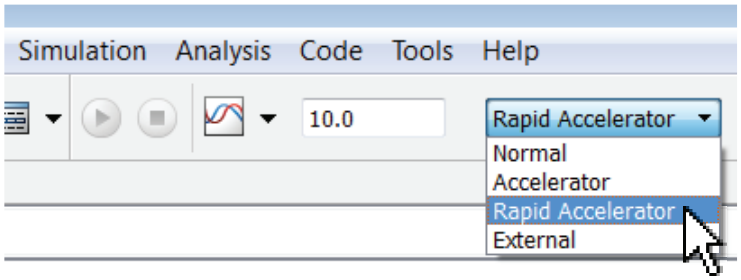
T-MATS Script

T-MATS Blocks

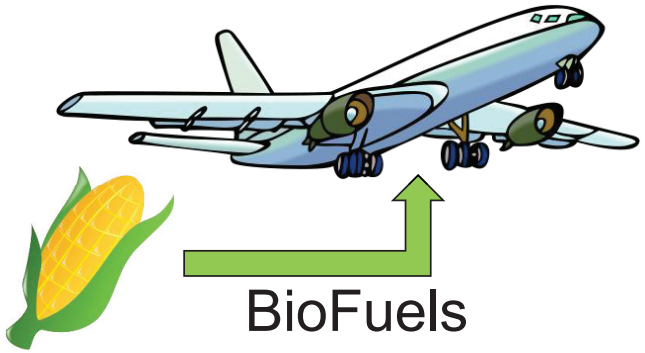


New Features: Enhancing capability

- Code generation
  - Generation of executables for operation outside of MATLAB environment or MATLAB accelerator modes



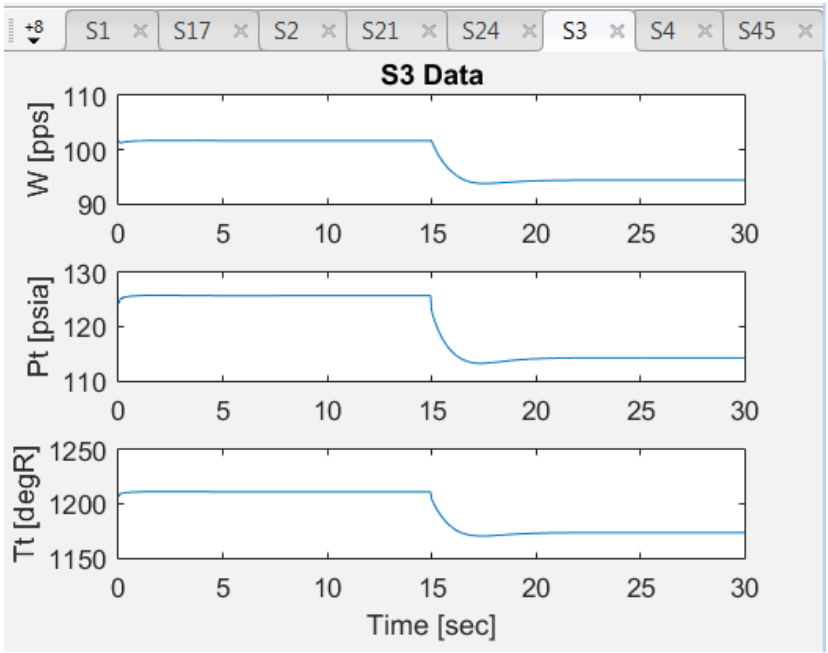
- Off Nominal Gas Property Tables
  - Create property tables to explore alternative fuels or air compositions with faster run times.



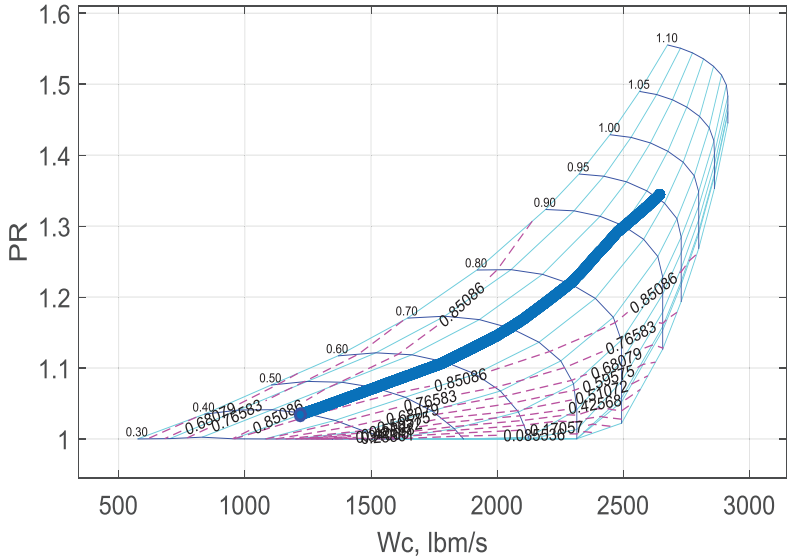
New Features: Visualization

- T-MATS plotting tools
  - Makes use of timeseries “To Workspace” blocks along with known output bus format to auto generate sets of plots to help to visualize engine performance.

Station Performance traces:



Dynamic map plotting:



Simple Syntax, after running the model use:

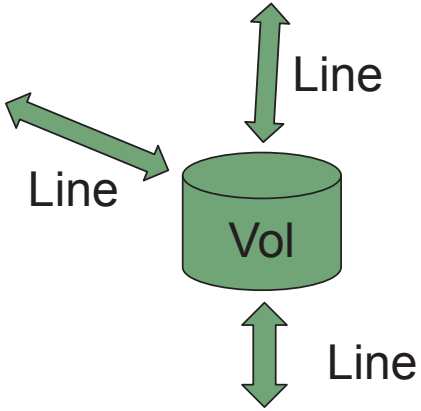
```
TMATS.TDplot('JT9D_Model_Dyn');
```

Additional Major Updates

- Engine heat soak dynamics
  - Add engine temperature effect to the simulation
  - Utilized lumped system approach

$$Q = m * C_p \dot{T}_{metal}$$

- Volume dynamics components

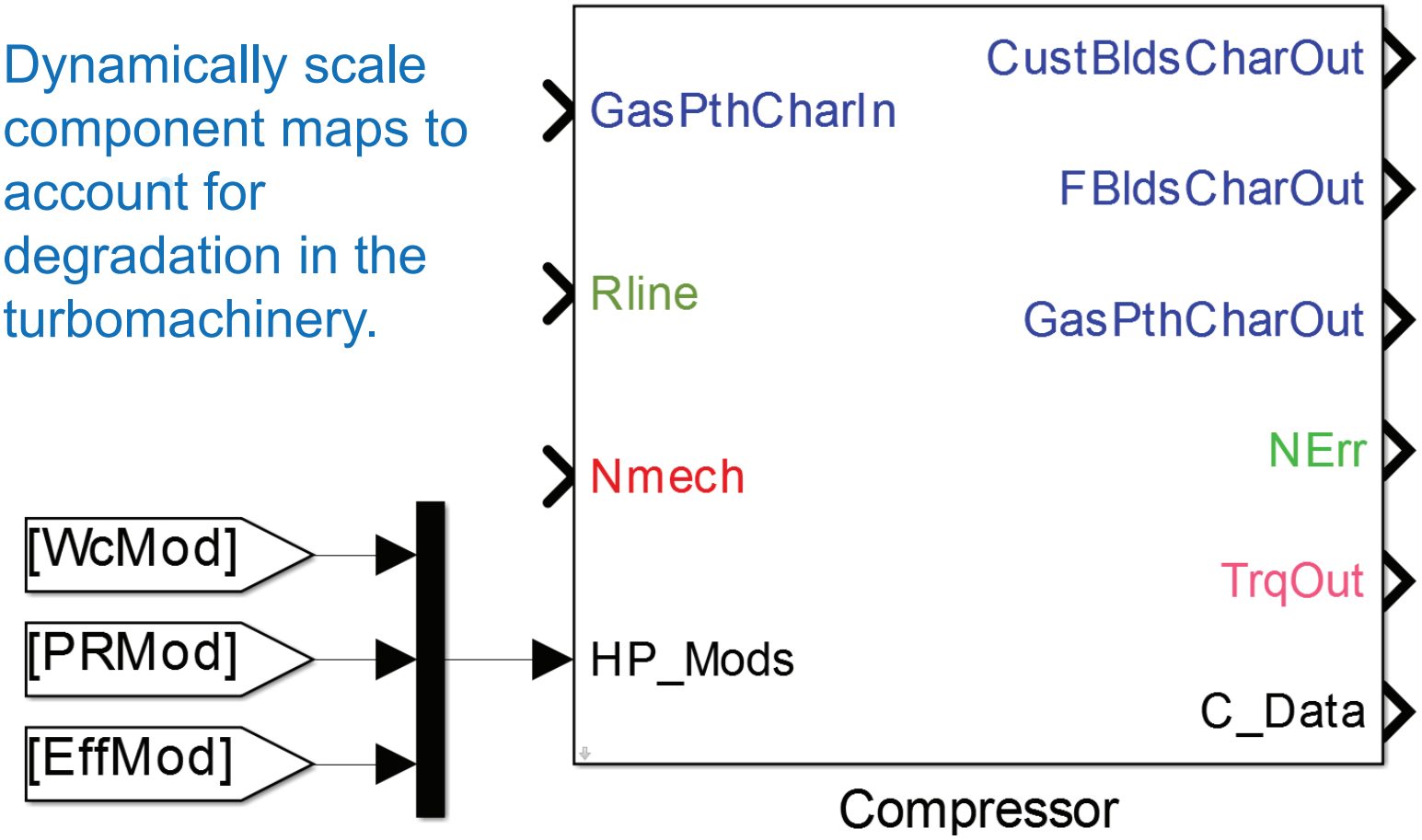


Solver used to converge the 1-D flow problem

- Generate upstream and downstream pressures and enthalpies
- Sum flows at volume node
- Converge to known density and internal energy.

Additional Major Updates

- Health parameter handles for turbomachinery
- Dynamically scale component maps to account for degradation in the turbomachinery.



Additional Major Updates

- Piecewise linear model creation
  - Utilizes perturbation method to generate linear models throughout a defined envelope.

$$\left(\dot{X}\right) = A(X - X_0) + B(U - U_0)$$
$$(Y - Y_0) = C(X - X_0) + D(U - U_0)$$

Where X is the system state, U is the system input, and Y is the plant output. 0 values are the trim point values. This block assumes U is dimension 1x1.

The simulation operates dynamically, pausing at each state to perform linearization.

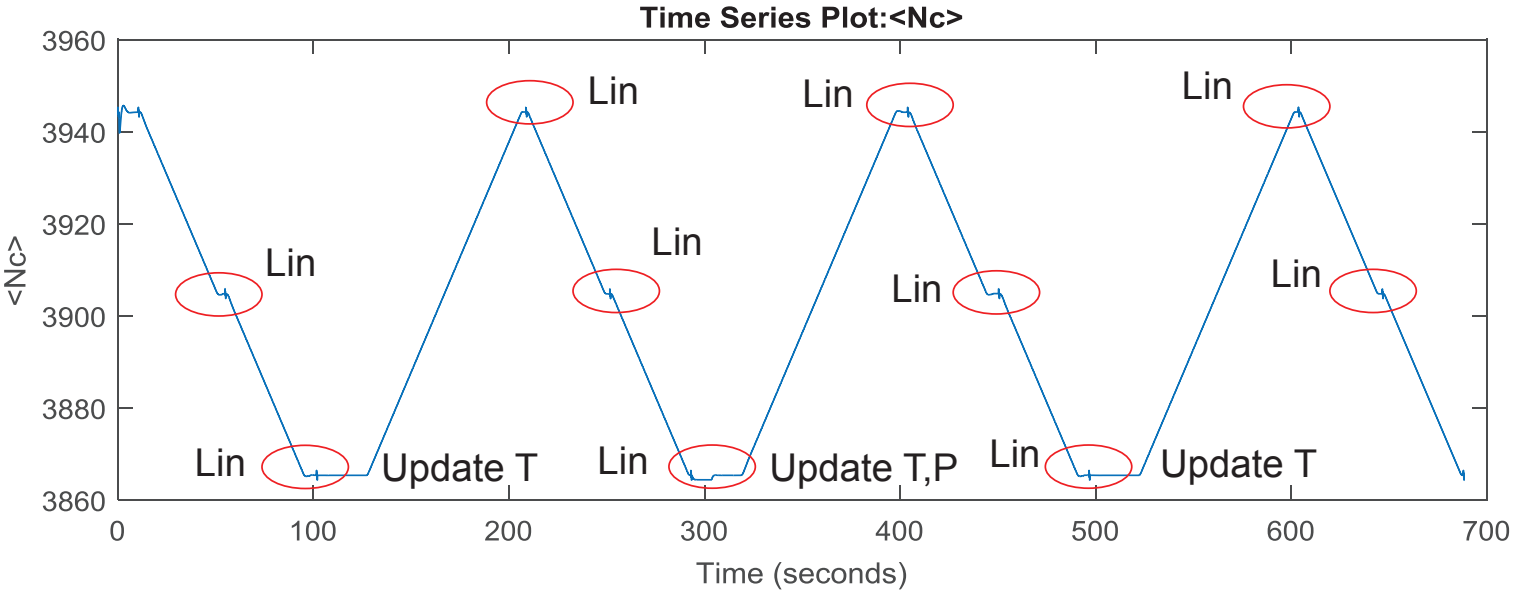
```
LinSys =

          ABCD: [1x1 struct]
        EnvVec: [1x1 timeseries]
             X: [1x1 timeseries]
             U: [1x1 timeseries]
             Y: [1x1 timeseries]
     CntrlSlew: [1x1 timeseries]
```

Output of the block is a .mat file that contains:

- ABCD matrix
- Matrix of non-input effectors
- X0,Y0, and U0 values
- Controller value, used to generate U

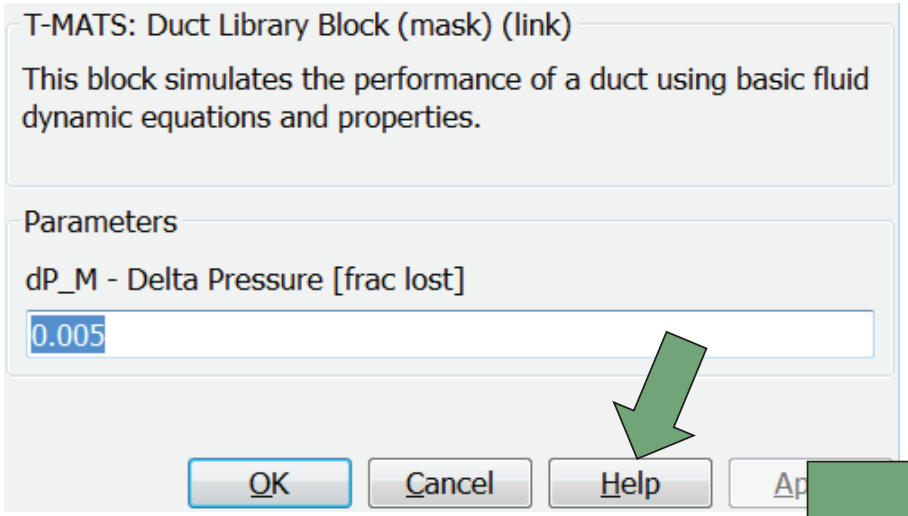
Additional Major Updates

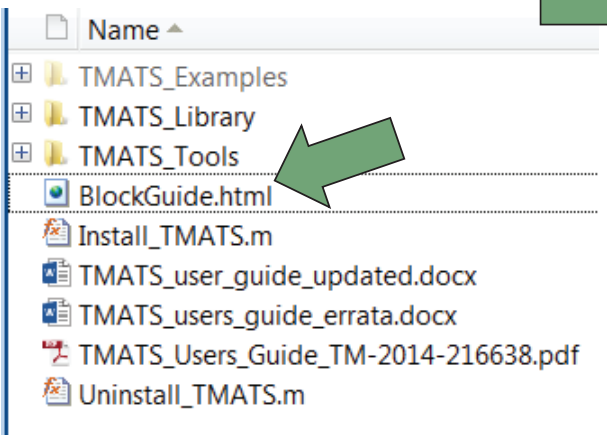
- Piecewise linear block example
  - Created to linearize the JT9D example in T-MATS
  - State space equation defined as U: Wf, X: Nf, N_hp
  - Control variable set to corrected fan speed (Nc below)
  - Input temperature and pressure are the two Env parameters
  - Linear models are generated at various Nc, T and P



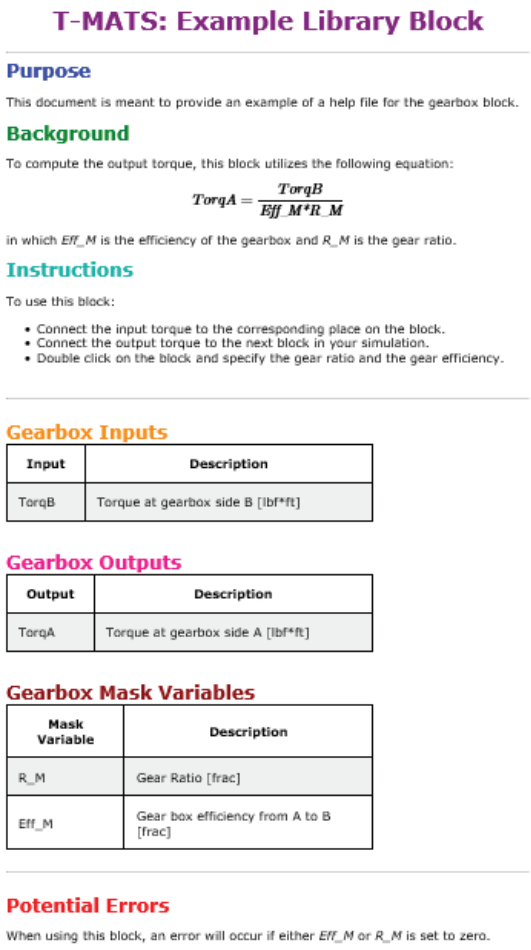
Help Files

Help files have been updated to be more…. Helpful.





Access through the Block Guide or by clicking on Help for any T-MATS or block



T-MATS: Example Library Block

Purpose

This document is meant to provide an example of a help file for the gearbox block.

Background

To compute the output torque, this block utilizes the following equation:

$$TorqA = \frac{TorqB}{Eff\_M * R\_M}$$

in which *Eff_M* is the efficiency of the gearbox and *R_M* is the gear ratio.

Instructions

To use this block:

- Connect the input torque to the corresponding place on the block.
- Connect the output torque to the next block in your simulation.
- Double click on the block and specify the gear ratio and the gear efficiency.

Gearbox Inputs

Input	Description
TorqB	Torque at gearbox side B [lbf*ft]

Gearbox Outputs

Output	Description
TorqA	Torque at gearbox side A [lbf*ft]

Gearbox Mask Variables

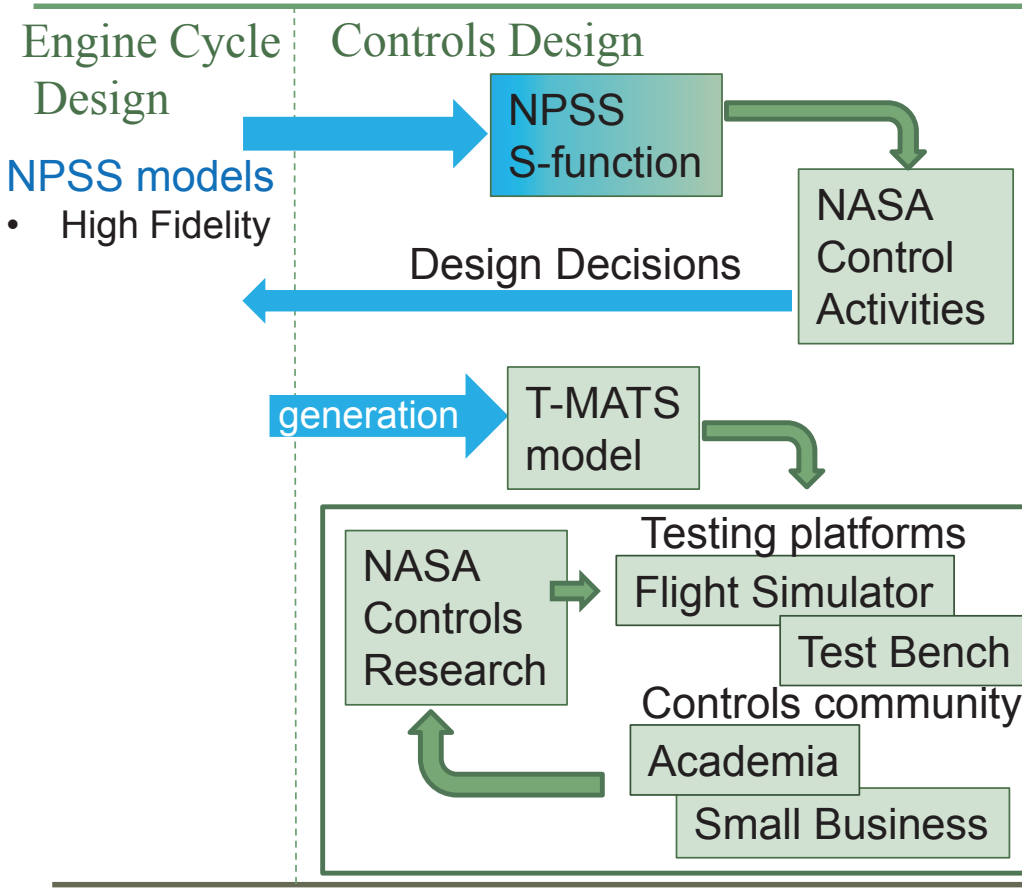
Mask Variable	Description
R_M	Gear Ratio [frac]
Eff_M	Gear box efficiency from A to B [frac]

Potential Errors

When using this block, an error will occur if either *Eff_M* or *R_M* is set to zero.

NPSS and T-MATS Relationship

- T-MATS works in harmony with and in parallel to industry standard engine modeling software, the Numerical Propulsion System Simulation (NPSS)
  - NPSS: Cycle design, truth models, high fidelity modeling
  - T-MATS: Controls design, fast development, fast hardware in the loop capability



S-function based design

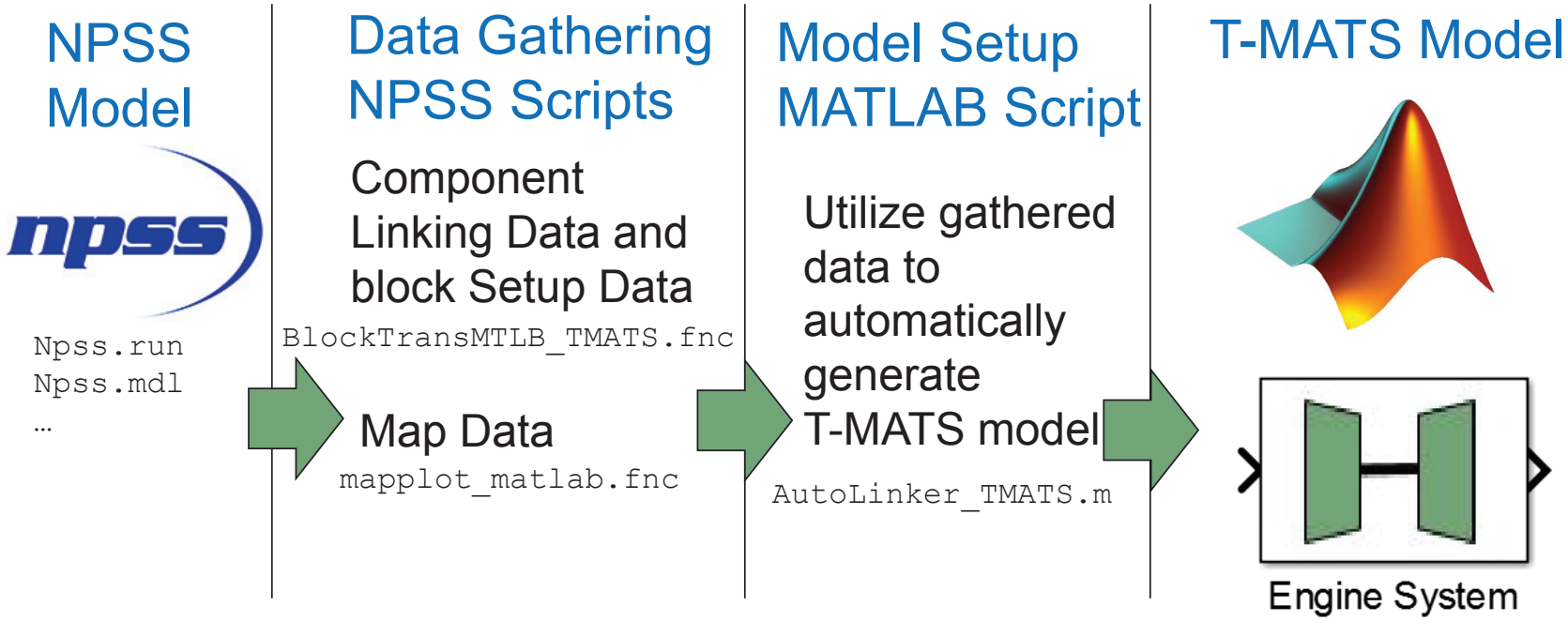
- Ideal for projects where multidisciplinary teams can collaborate
- Exact model match with truth model
- Promotes rapid prototyping between engine cycle design and controls

T-MATS based research

- Allows conceptual designs to be quickly brought to testing platforms
- Needs based model fidelity
- Enables controls research using a single tool
- Promotes aero propulsion to engineers without NPSS experience
- Allows independent research activities with the controls community

New Features: Model Auto-Generation

- NPSS to T-MATS auto coder
  - Utilizes a like-for-like building process to generate a T-MATS model directly from an NPSS model.



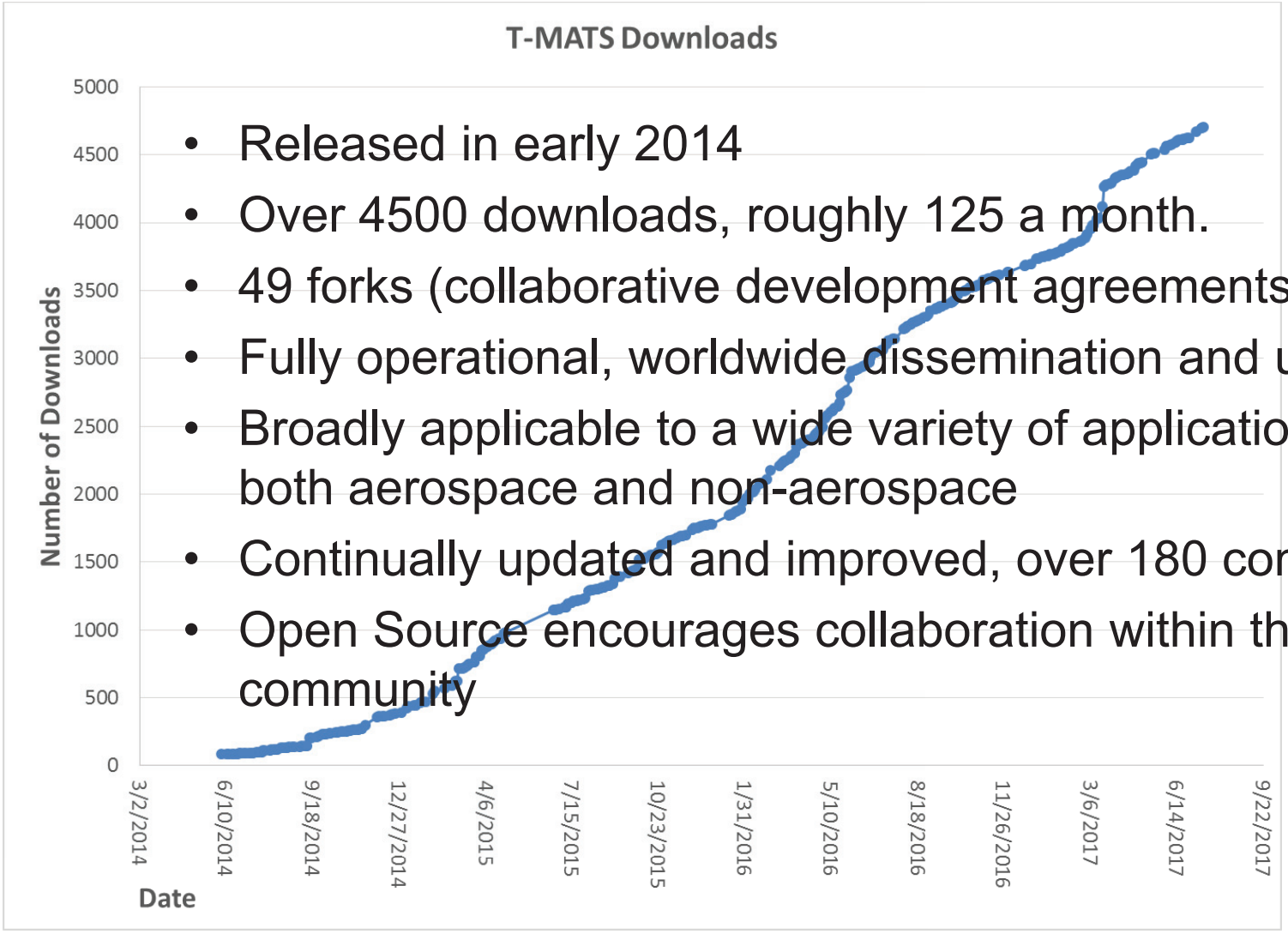
NASA/CP—2018-219785

NASA T-MATS usage

- Major NASA activities that make use of T-MATS
  - Aircraft Engine Icing Detection and Mitigation
    - Developed a turbofan engine model that captures system-level effects of ice particle ingestion and ice blockage in the compression system under closed-loop control operation
  - Gas Electric propulsion for Civilian Commuter Operations (GECCO)
    - Ongoing project to develop a small turbogenerator (turboshaft + generator) propulsion system model for small commuter aircraft.
  - Distributed Engine Control (DEC)
    - Modified and built a geared turbofan model to operate within the distributed engine controls rig.
  - Active Turbine Tip Clearance Control (ATTCC)
    - Created an engine model that simulates the mechanical growth of components relating to turbine tip clearance. Integrated this tip clearance model with an advanced geared turbofan.
  - Model Based Engine Control (MBEC)
    - Ongoing work to develop an engine model for validation of MBEC algorithms before they are implemented within the test cell.

NASA/CP—2018-219785

Status



- Released in early 2014
- Over 4500 downloads, roughly 125 a month.
- 49 forks (collaborative development agreements)
- Fully operational, worldwide dissemination and use
- Broadly applicable to a wide variety of applications, both aerospace and non-aerospace
- Continually updated and improved, over 180 commits
- Open Source encourages collaboration within the community

Summary

- T-MATS offers a comprehensive thermodynamic simulation system
  - Major updates include NPSS model translation, data visualization, and platform compatibility.
  - Increased engine modeling functionality ranging from health parameters to heat soak
  - T-MATS can be downloaded at the address:

    https://github.com/nasa/T-MATS/releases
  - Write to the community at the T-MATS user's forum:

    https://groups.google.com/forum/#!forum/t-mats-user-group

NASA/CP—2018-219785

Acknowledgments

Funding for this work was provided by the Transformative Aeronautics Concepts Program (TACP)/Transformational Tools and Technologies (TTT) project.

NASA/CP—2018-219785

NASA/CP—2018-219785

T-MATS Volumetric Blocks

Aidan W. Rinehart
Vantage Partners LLC

T-MATS Workshop
August 21, 2017
Cleveland, Ohio

Outline

- Modeling Fundamentals
- Currently Developed Blocks
- Simple Servo-valve Dual Action Piston Example
- Future Development
- Conclusions

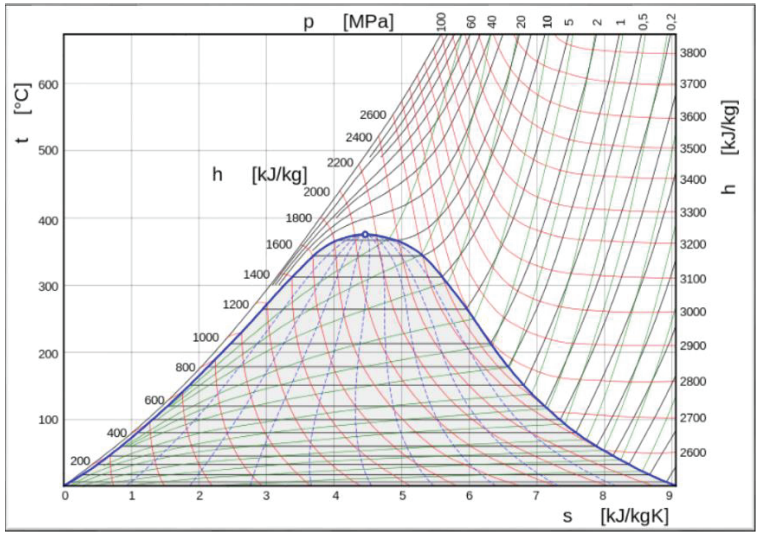
NASA/CP—2018-219785

Objective

- Develop fundamental blocks within the T-MATS environment that can model working fluid systems that are capable of capturing the dynamic responses of the fluid and associated mechanical systems. Including but not limited to hydraulic actuators and high pressure rocket lines.

Fundamental Modeling Principles

- Fluid property tables developed through National Institute of Standards and Technology (NIST) program RefProp
- Refprop utilizes Helmholtz equations of state to calculate working fluid properties
- 2D property lookup tables built using Refprop for working fluid



'Example T-s diagram' Kaboldy (Own work) [CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons

Fundamental Modeling Principles

- Volumetric blocks solve for working fluid properties through property lookup tables
- Independent variables:
  - Pressure
  - Enthalpy
  - Temperature*
- Dependent variables:
  - Density
  - Internal energy
  - Temperature

- Density and internal energy are both calculated and recalled from lookup tables
- The difference between the two methods provides the error term used to drive the solution

*Lookup tables can be designed to use any two properties. Pt-ht and Pt-Tt have been used as the test cases

Flow Start

- Inputs:
  - Mass flow rate
  - Inlet temperature
  - Inlet pressure
- Outputs
  - Outlet flow properties
  - Fluid properties

- Look up properties of working fluid based on inlet pressure and temperature
- Initiates flow conditions based on user supplied inputs

Constant Volume

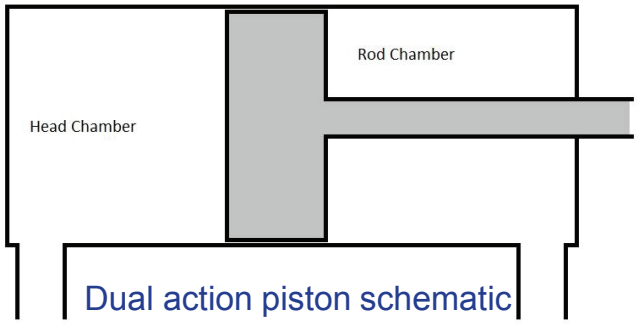
- Inputs:
  - Volume
  - Inlet flow properties
  - Downstream flow properties
  - Enthalpy
  - Integrated density
  - Integrated internal energy
- Outputs
  - Outlet flow properties
  - Change in density
  - Change in internal energy
  - Density error
  - Internal energy error

- Fixed volume
- Supports multiple in and out flows
- Calculates instantaneous change in density and internal energy
- Looks up density and internal energy based on pressure and enthalpy
- Pressure and enthalpy varied until difference between integrated and lookup values are within solution error limits

NASA/CP—2018-219785

Variable Volume

- Inputs:
  - Volume
  - Inlet flow properties
  - Outlet flow properties
  - Enthalpy
  - Integrated density
  - Integrated internal energy
  - Integrated mass
- Outputs
  - Outlet flow properties
  - Change in density
  - Change in internal energy
  - Pressure error
  - Internal energy error



Dual action piston schematic

- Supports multiple in and out flows
- Calculates instantaneous change in pressure and internal energy
- Looks up internal energy based on pressure and enthalpy
- Pressure and enthalpy varied until difference between integrated and lookup values are within solution error limits
- Internal pressure calculated based on Redlich-Kwong equation of state

$$a = \frac{0.42748\bar{R}^2 T_c^{\frac{5}{2}}}{P_c} \qquad \mathrm{b} = \frac{0.08664\bar{R}T_c}{P_c}$$

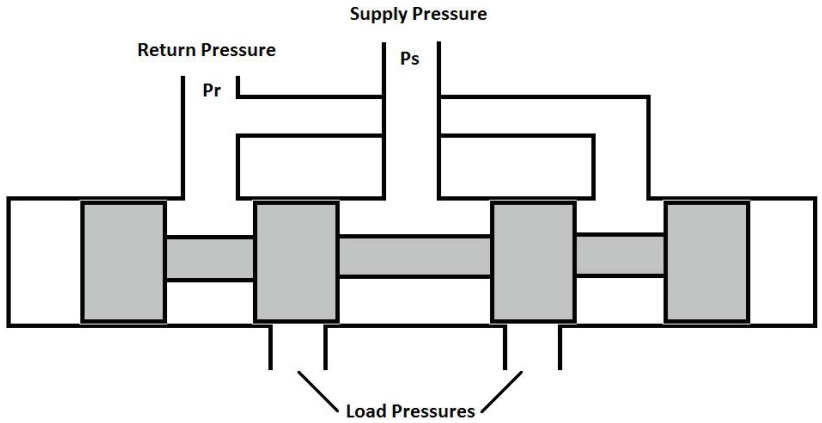
$$P = \frac{\bar{R}T}{\bar{v} - b} - \frac{a}{\bar{v}(\bar{v} - b)T^{0.5}}$$

Valve

- Inputs:
  - Inlet flow properties
  - Down stream pressure
  - Down stream enthalpy
  - Cross sectional area open
  - Maximum cross sectional area
  - Discharge coefficient
- Outputs
  - Outlet flow properties
  - Down stream enthalpy

$$W = Cd * Ac * \rho \sqrt{\frac{2(P_1 - P_2)}{\rho}}$$

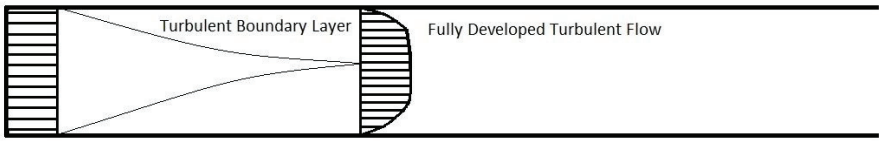
- Variable cross section area
- The pressure differential determines the flow direction
- Discharge coefficient determines the losses associated with orifice shape
- Auxiliary function provides current cross sectional area that is open
- 4-land spool is a valve system used to control flow to actuator systems



Schematic of 4 land spool valve

Pipe

- Inputs:
  - Diameter
  - Length
  - Pressure differential
  - Inlet density and viscosity
  - Outlet density and viscosity



Fully developed turbulent pipe flow schematic

- Outputs
  - Mass flow rate
  - Enthalpy
  - Total temperature
  - Total pressure

- Circular cross section
- Fully developed laminar and turbulent mass flow rates calculated
- Flow is considered laminar when $R_{laminar} < 2000$ and turbulent for $R_{laminar} \geq 2000$

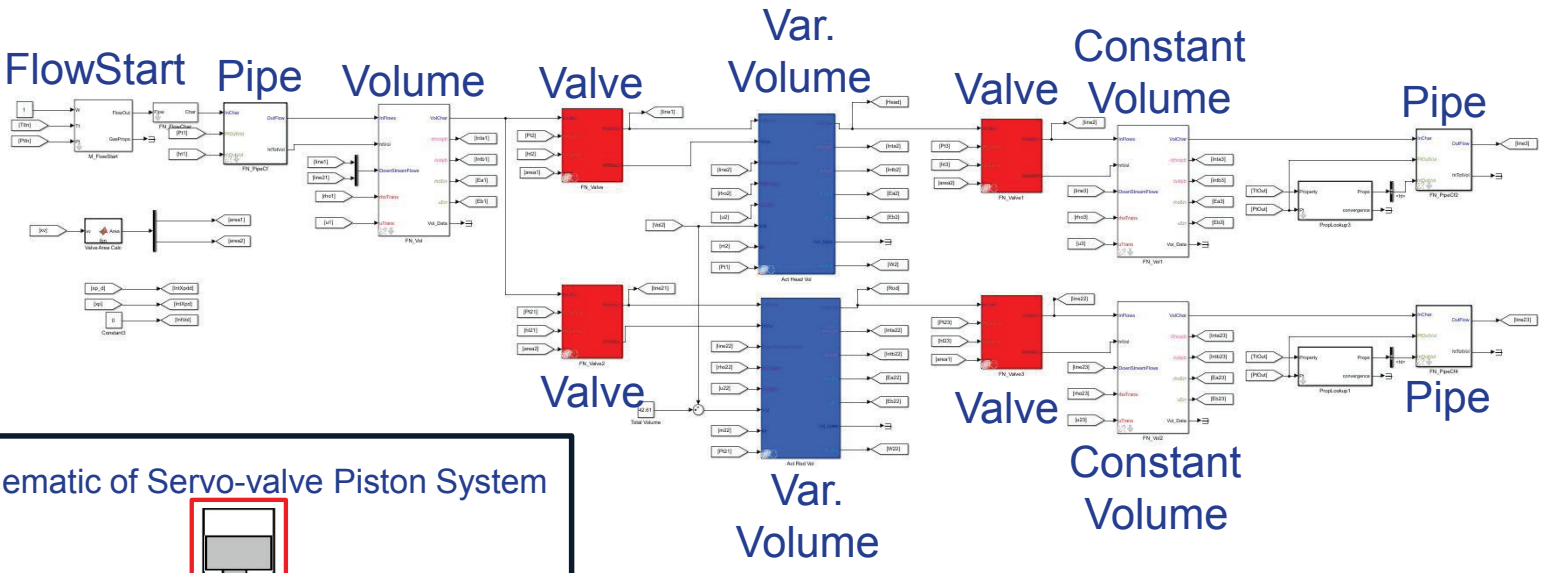
$$W_{laminar} = \frac{(P_1 - P_2)\rho\pi\left(\frac{D}{2}\right)^4}{8\mu L}$$

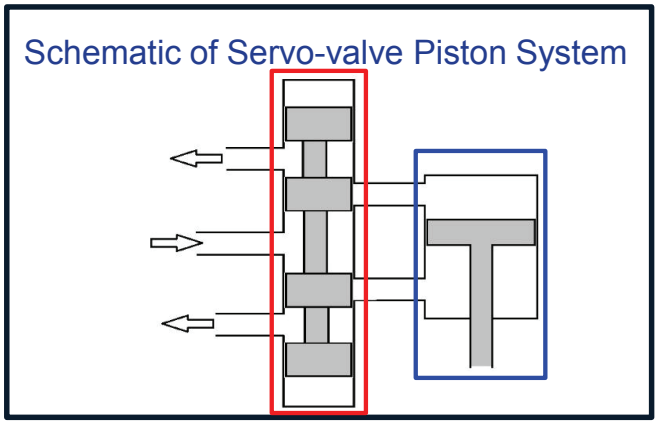
$$W_{turbulent} = \frac{\rho(P_1 - P_2)D^{4.75}}{(0.242L\mu^{0.25}\rho^{0.75})^{1/1.75}}$$

$$R_{laminar} = \frac{4W_{laminar}}{\pi D\mu}$$

Example

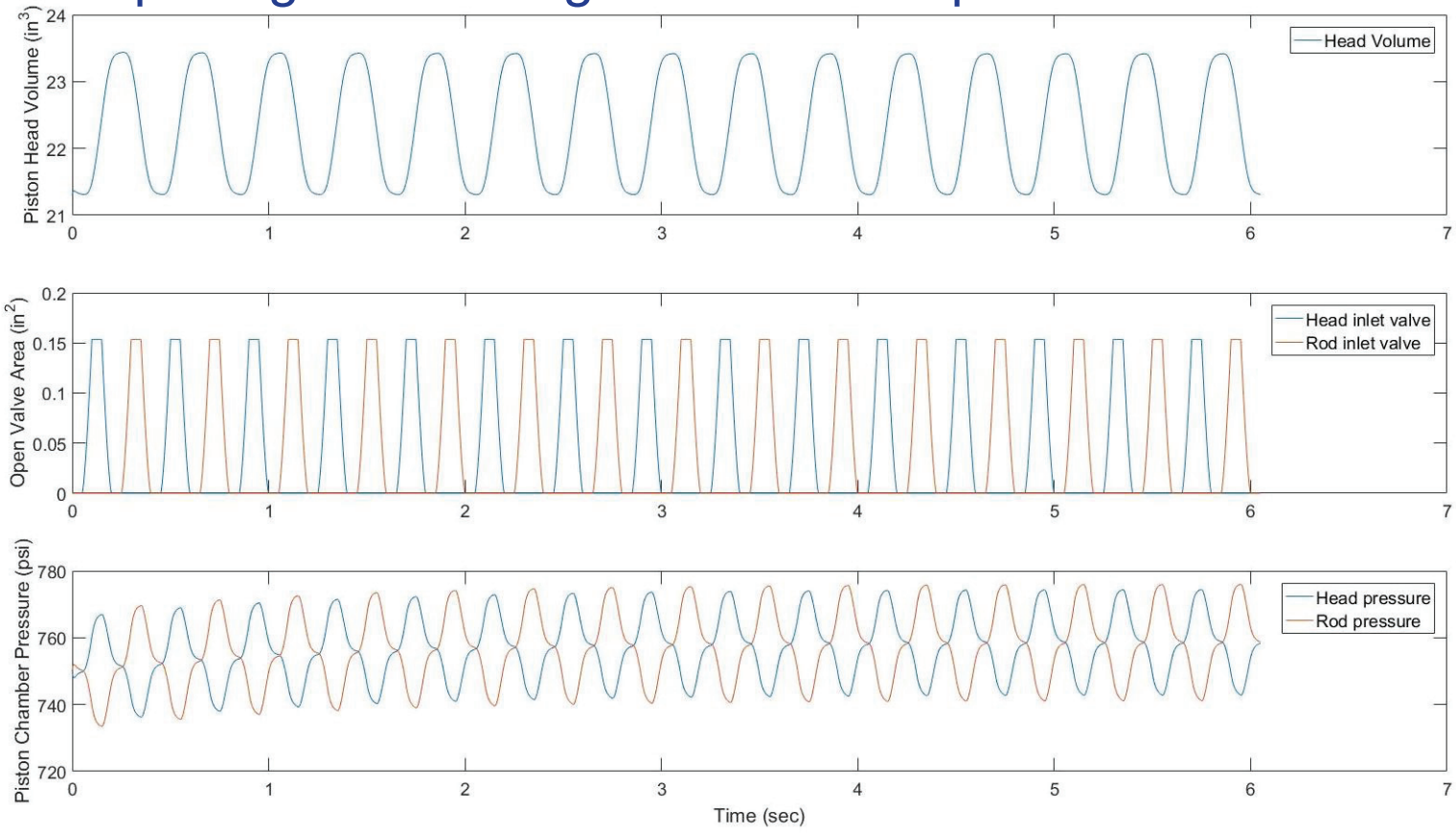
- Simple 4 land servo valve attached to a dual action piston





Results

- Modulation of servo valve through 15 cycles of opening and closing head and rod piston chambers



NASA/CP—2018-219785

Future Work

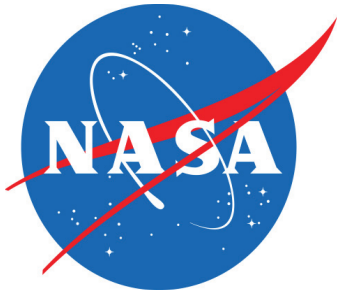
- Develop more fundamental blocks to enable wider range of modeling
- Compare results with other modeling tools
- Investigate revisions to speed up simulation run time

Summary

- Established a modeling technique for capturing the dynamics of working fluids
- Developed several fundamental block elements that can be combined to model working fluid systems

Acknowledgments

This work was conducted under the Transformative Aeronautics Concepts Program (TACP)/Transformational Tools and Technologies (TTT) project.



References

- Chapman, Jeffryes W., Lavelle, Thomas M., May, Ryan D., Litt, Jonathan S., Guo, Ten-Huei, “Toolbox for the Modeling and Analysis of Thermodynamic Systems (TMATS) User’s Guide,” NASA/TM—2014-216638, January 2014.
- Chapman, Jeffryes W., Lavelle, Thomas M., May, Ryan D., Litt, Jonathan S., Guo, Ten-Huei, “Propulsion System Simulation Using the Toolbox for the Modeling and Analysis of Thermodynamic Systems (T-MATS),” AIAA 2014-3929, 50th AIAA/ASME/SAE/ASEE Joint Propulsion Conference, Cleveland, OH, July 28-30, 2014, also NASA/TM—2014-218410, November 2014.
- Zinnecker, Alicia M., Chapman, Jeffryes W., Lavelle, Thomas M., Litt, Jonathan S., “Development of a twin-spool turbofan engine simulation using the Toolbox for Modeling and Analysis of Thermodynamic Systems (T-MATS),” AIAA 2014-3930, 50th AIAA/ASME/SAE/ASEE Joint Propulsion Conference, Cleveland, OH, July 28- 30, 2014, also NASA/TM—2014-218402, November 2014.
- Chapman, Jeffryes W., Lavelle, Thomas M., Litt, Jonathan S., Guo, Ten-Huei, “A Process for the Creation of T-MATS Propulsion System Models from NPSS data,” AIAA 2014-3931, 50th AIAA/ASME/SAE/ASEE Joint Propulsion Conference, Cleveland, OH, July 28-30, 2014, also NASA/TM—2014-218409, November 2014.
- Lavelle, Thomas M., Chapman, Jeffryes W., May, Ryan D., Litt, Jonathan S., Guo, Ten-Huei, “Cantera Integration with the Modeling and Analysis of Thermodynamic Systems (T-MATS)," AIAA 2014-3932, 50th AIAA/ASME/SAE/ASEE Joint Propulsion Conference, Cleveland, OH, July 28-30, 2014.

NASA/CP—2018-219785

General use Geared Turbofan Simulation in T-MATS

Jeffryes W. Chapman, Vantage Partners, LLC.

2nd T-MATS Workshop
Ohio Aerospace Institute (OAI)
Cleveland, OH
August 21, 2017

Background

- In preparation for the next generation of aircraft, T-MATS has been used to model advanced high-efficiency engine concepts.

  The Advanced Geared Turbofan, 30,000 lbf (AGTF30) engine simulation was developed to investigate possible next generation engine system designs including:

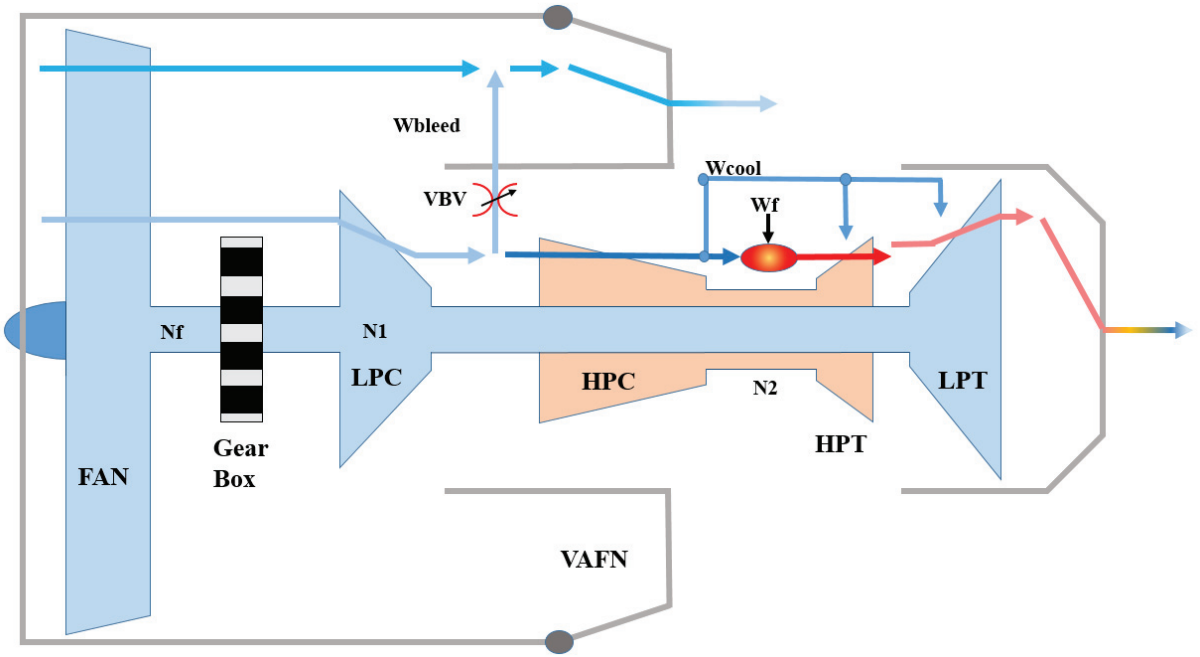
  1. Dual spool Geared Turbofan engine design
  2. Ultra-high bypass configuration
  3. Small engine core
  4. Variable area fan nozzle (VAFN)
  5. Fully operational dynamic control system

- Purpose
  - Provide a dynamic platform for next generation engine system research.

NASA/CP—2018-219785

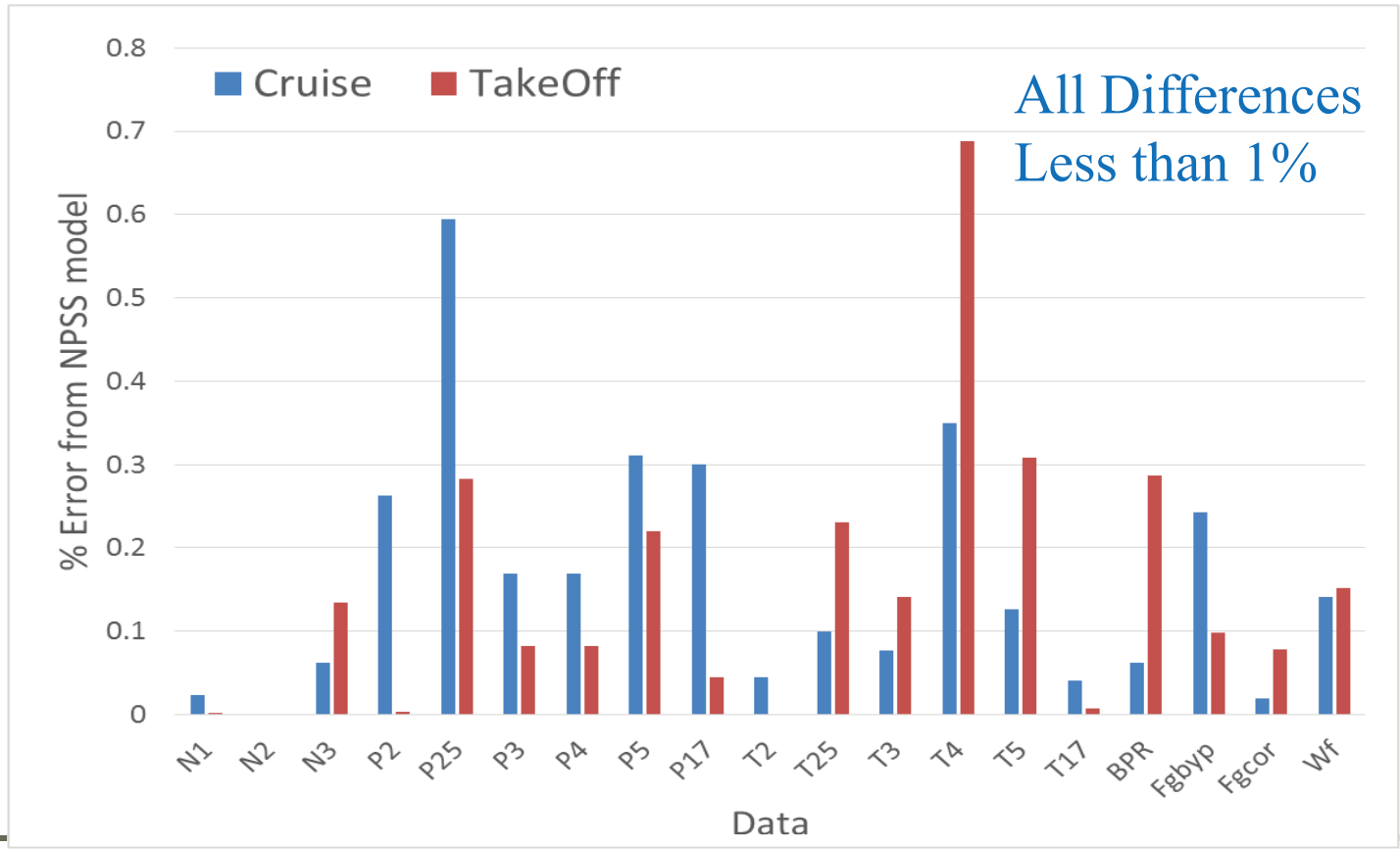
Engine Model Description

- Advanced Geared Turbofan features
  - Variable area fan nozzle (VAFN)
  - Dual spool with low pressure shaft connected to fan via a gear box
- Performance
  - BPR = 24, OPR = 50, TIT = 3000, TSFC = 0.46 at cruise
  - 30,000 lbf takeoff thrust
- Control Effectors: VAFN, fuel flow (Wf), and variable bleed valve (VBV)



Validation, Design Point

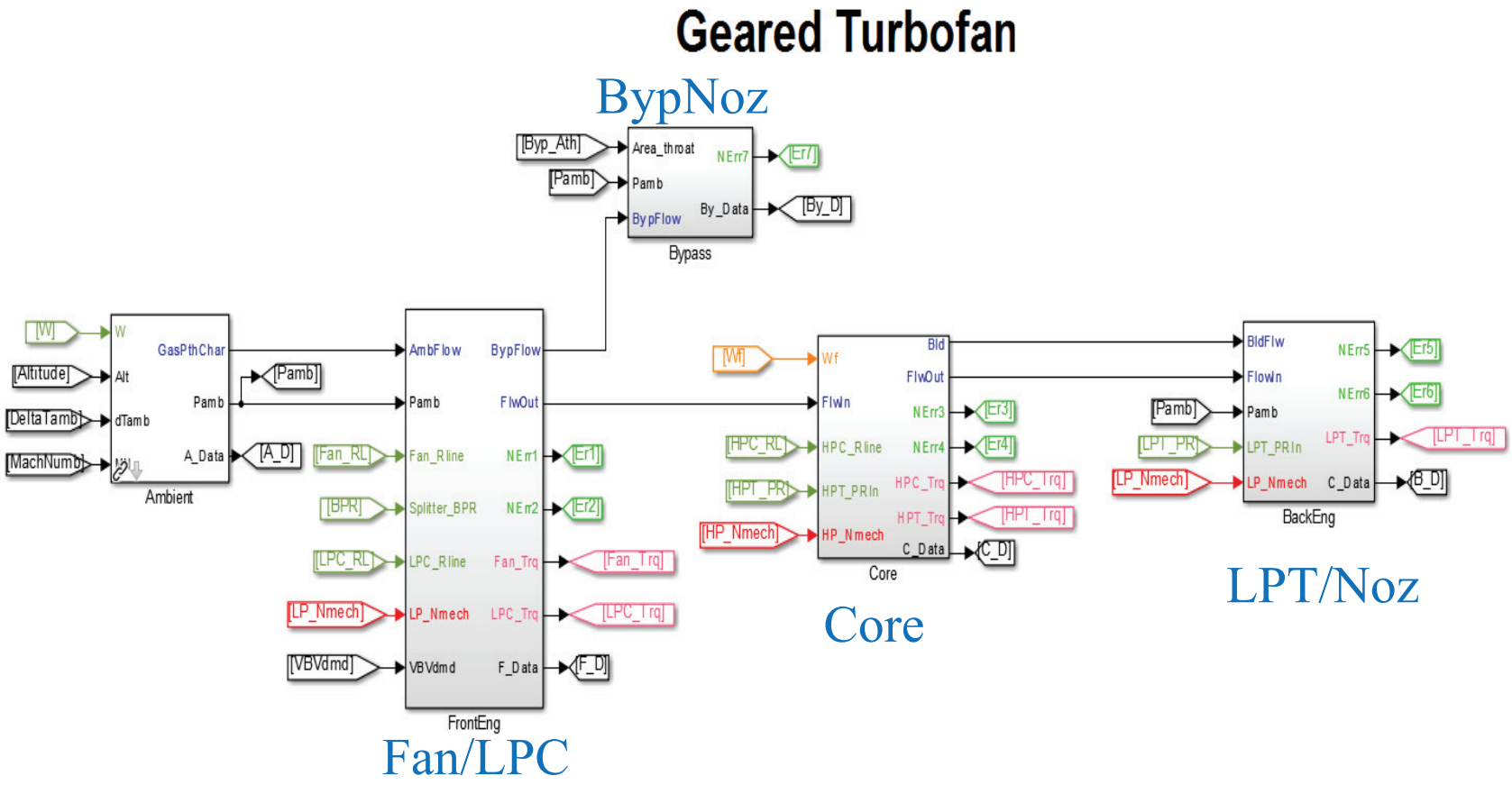
- AGTF30 model, based on NPSS cycle model
  - 2 operational points compared: Cruise and Take off
    - Utilizes steady-state solving techniques with a known VAFN area



NASA/CP—2018-219785

AGTF30 engine, model

- T-MATS Engine model



Expected Deviations, Off Design

- Several assumptions were made in the NPSS cycle model that are not practical for a controls development model
  - VAFN converged to a Fan Op-line
  - Bleed air set to maintain low pressure compressor(LPC) stability margin (SM) > 10
  - NPSS Fan map range does not extend to idle
- Planned Adjustments for T-MATS model
  - VAFN scheduled based on corrected fan speed
  - VBV added with position scheduled based corrected fan speed
  - NPSS Fan map was extended to reach idle values
- Other items to be aware of:
  - The NPSS model calculates stability margin (SM) based on map value (unscaled), this results in a SM different than SMavail (based on scaled map). To maintain matching with the NPSS model, it is suggested to use SMmap for research purposes.

Modeling to Match

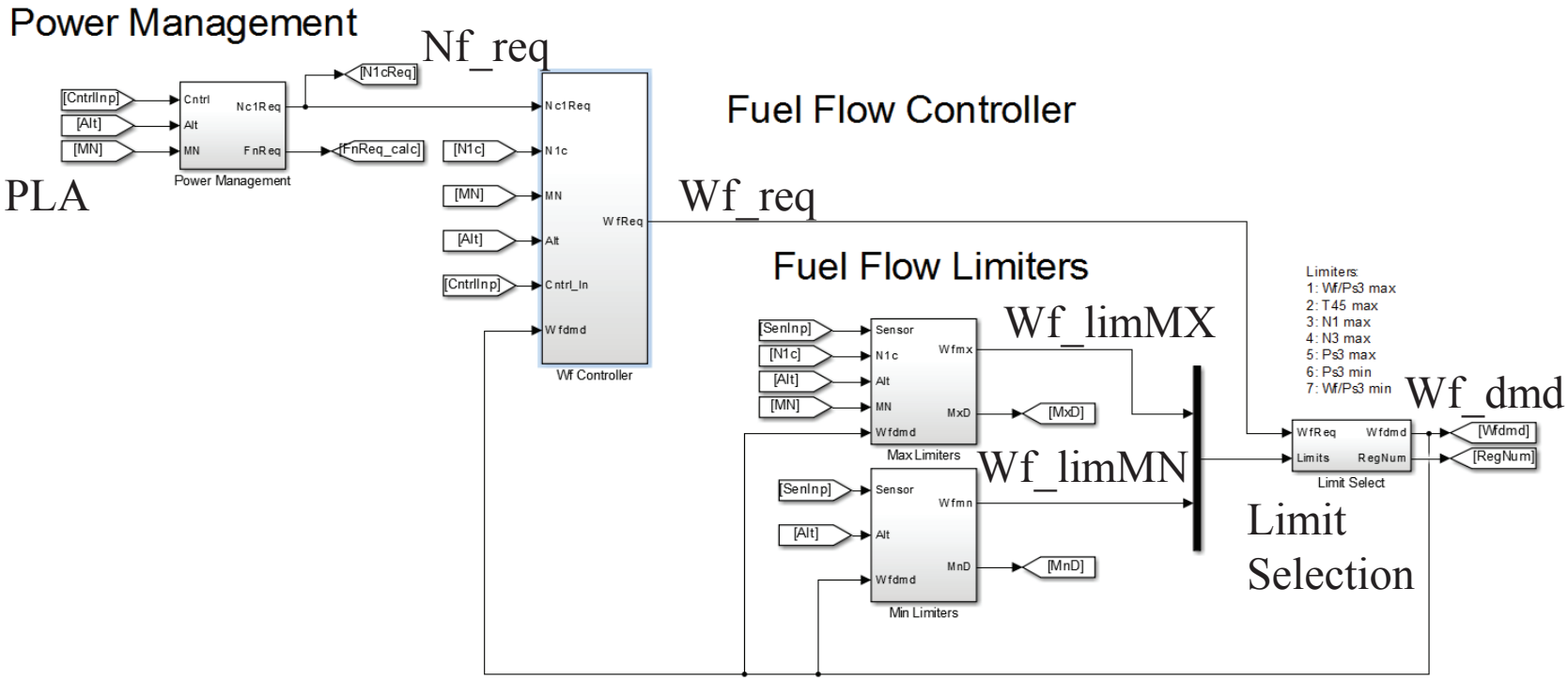
- To gain a better matching with the NPSS model several T-MATS blocks were modified
  - NPSS duct model scales dP by normalized MN

T-MATS default	GTF NPSS model
$dP = constant$	$dP = dPdes \left(\frac{MN}{MNdes}\right)^2$

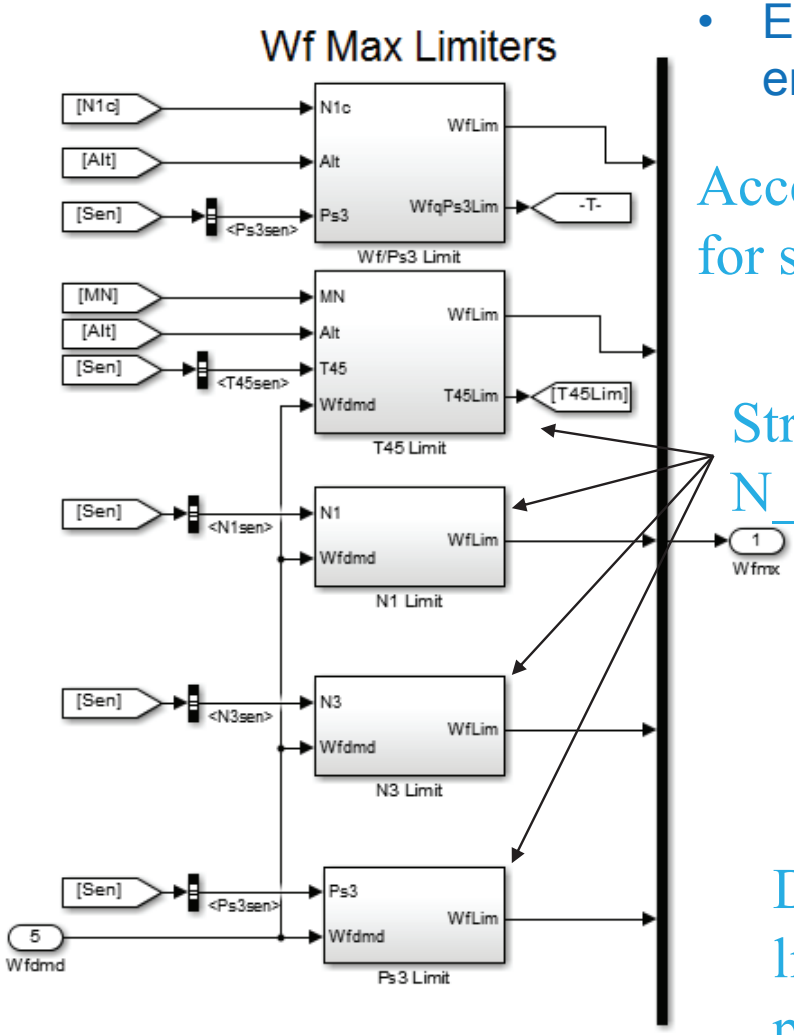
  - Gear box efficiency was applied directly to LPT torque

Fuel Control Architecture

- Fuel Control methodology based on literature
  - Power Management generates fan speed request based on power lever angle (PLA)
  - Fan speed controller generates a fuel flow request
  - Sets of limiters adjust the fuel flow request to operate the engine safely; avoiding engine stall, structural limits, combustor blowout, etc.
  - Controllers utilize PI method, tuned to meet requirements throughout the envelope



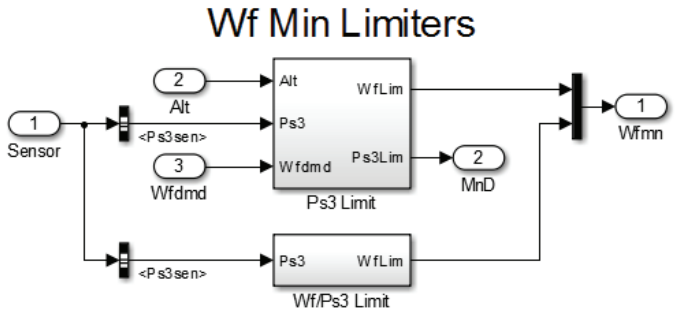
Fuel Control Architecture



- Each fuel limiter designed to protect the engine.

Acceleration limit
for stall margin mitigation, Wf/Ps3

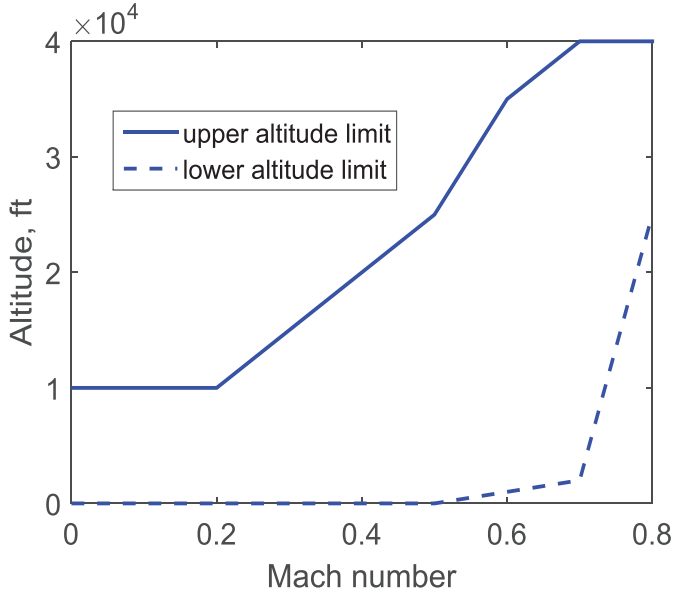
Structural limits, T45,
N_fan, N_hp, Ps3



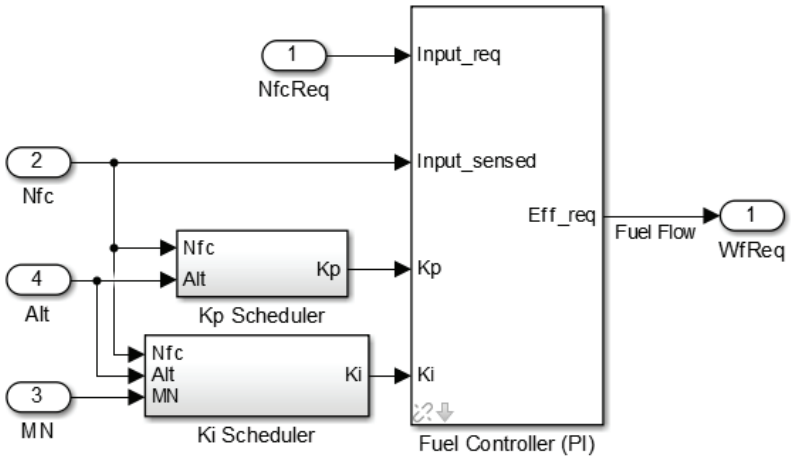
Deceleration and Pressure
limits for combustor blow out
protection, Wf/Ps3, Ps3

Fuel Control Tuning

- PI controller gains tuned to ideal values throughout envelope
  - Linear models were generated throughout the envelope and at various power levels
  - PI controller gains were tuned for each defined linear model.
  - Gains were collected into schedules that provide the optimum gain at each operational point.



Operational Envelope

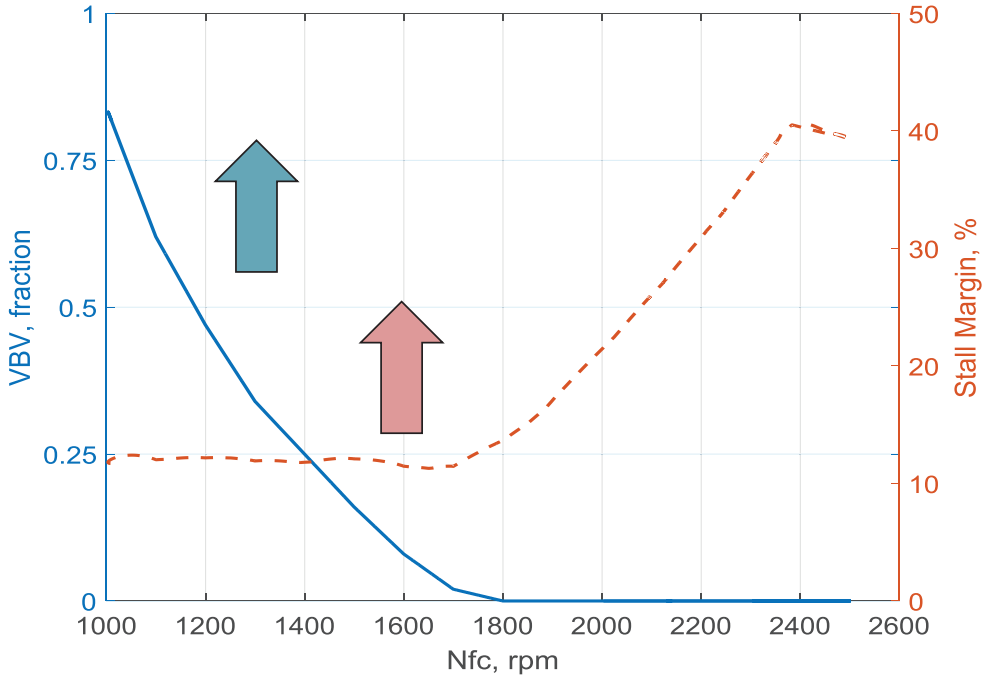


Speed Controller

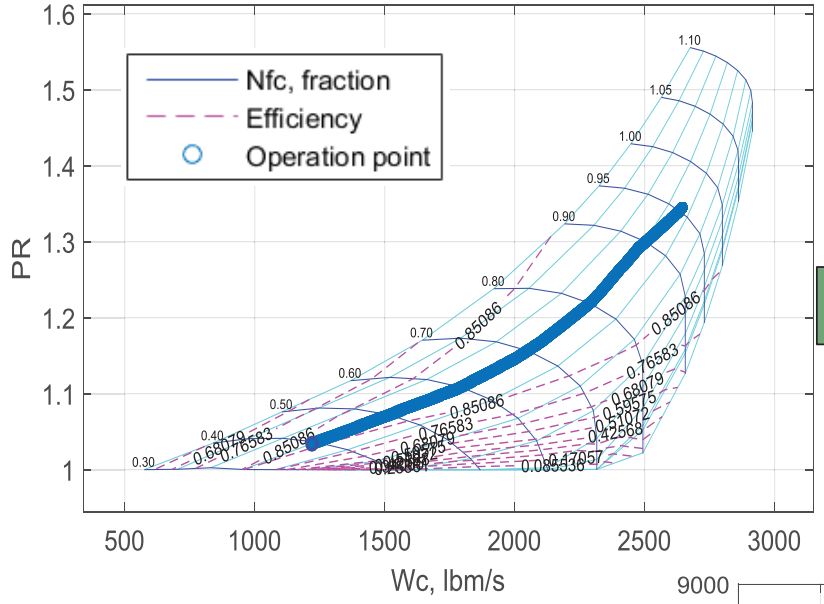
VBV Control Architecture

- Variable bleed valve opens to reduce low pressure compressor (LPC) pressure ratio (PR), increasing stall margin.
  - Schedules constructed to maintain 10% stall margin during steady-state operation.

Opening VBV to increase LPC stall margin



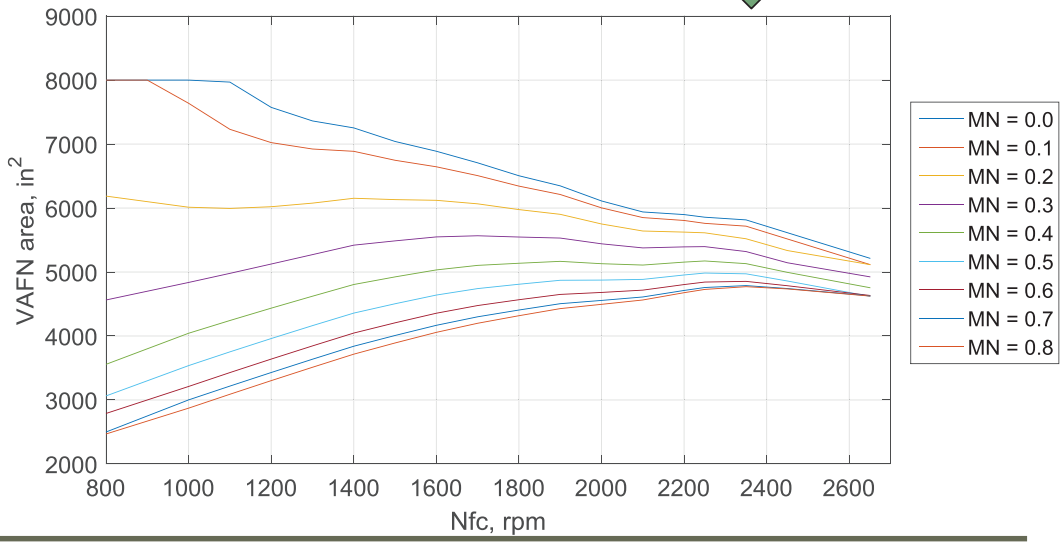
VAFN Control Architecture



Fan Performance

Optimal efficiency

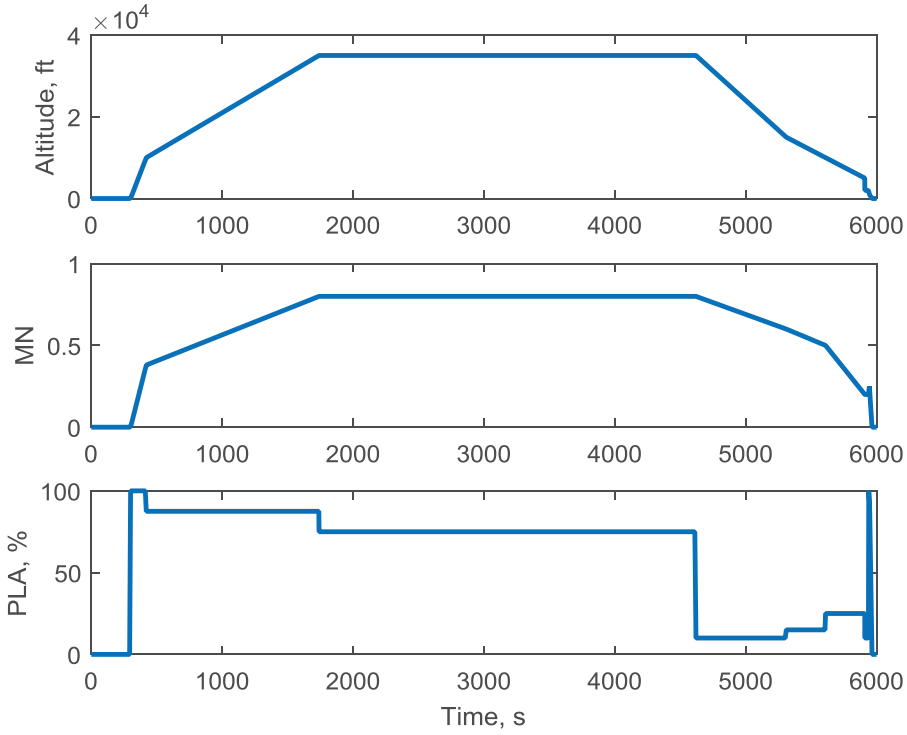
VAFN Schedule



- Variable area fan nozzle area scheduled to maintain optimal fan efficiency.
  - Nozzle area increased to reduce fan PR
  - Nozzle area decreased to increase fan PR

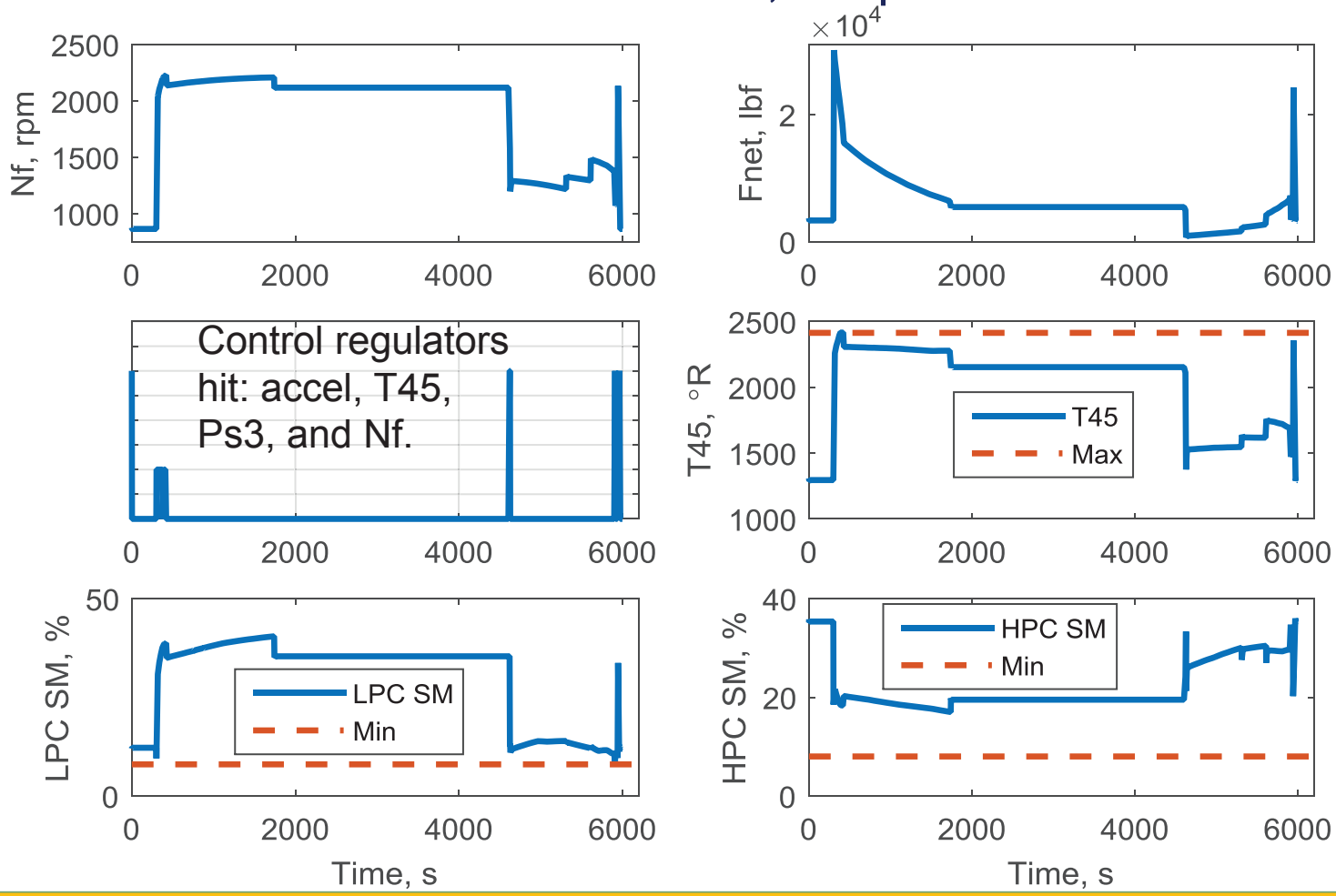
Model Validation

- Engine Model validation
  - Simulation of an abbreviated mission profile
    - Engine idling
    - Acceleration from idle to full power followed by a take off at sea level static conditions
    - Engine climbs to cruise at 35,000 ft
    - Deceleration and descent
    - Aircraft lands then returns to idle



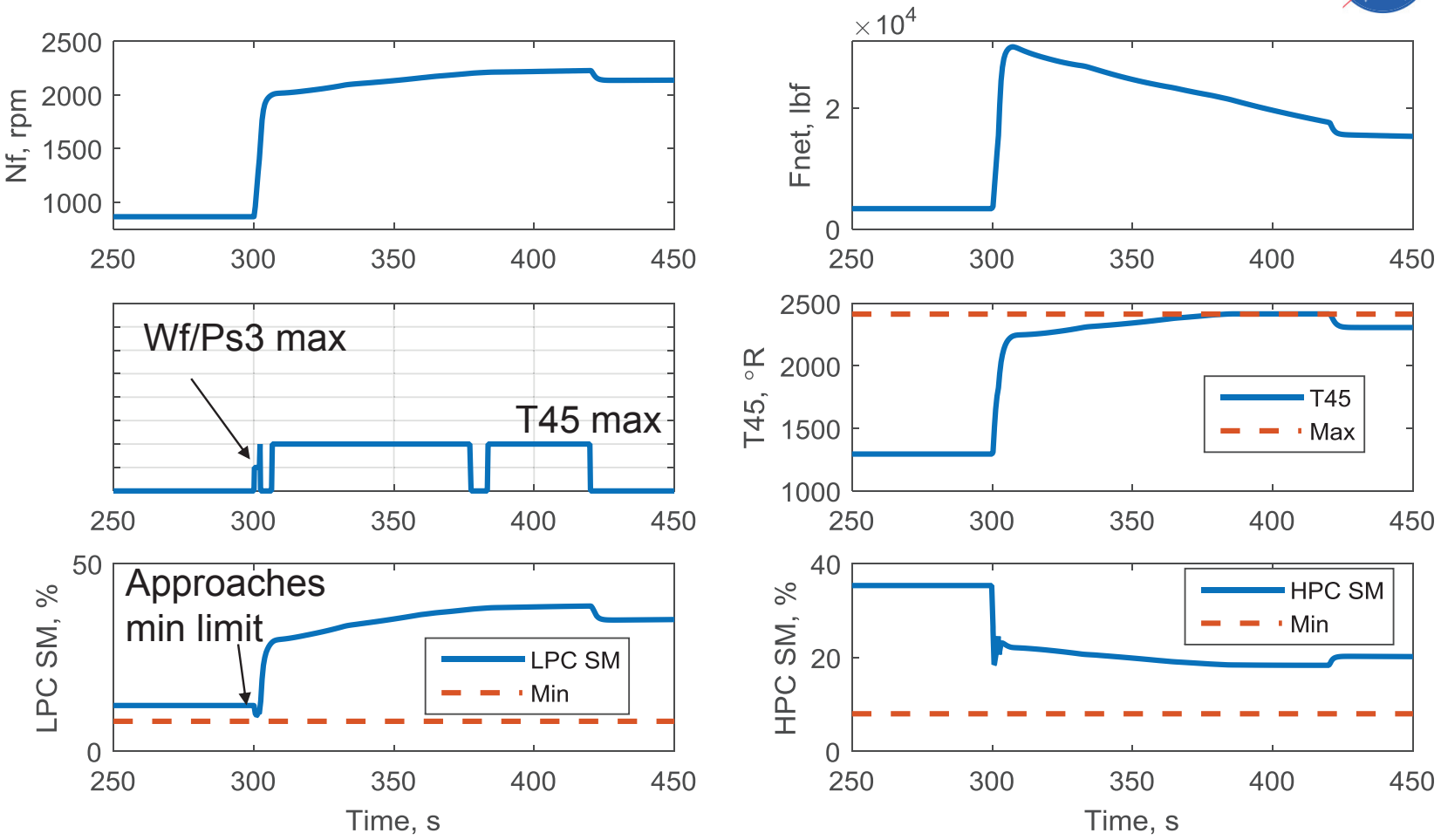
NASA/CP—2018-219785

Model Validation, full profile



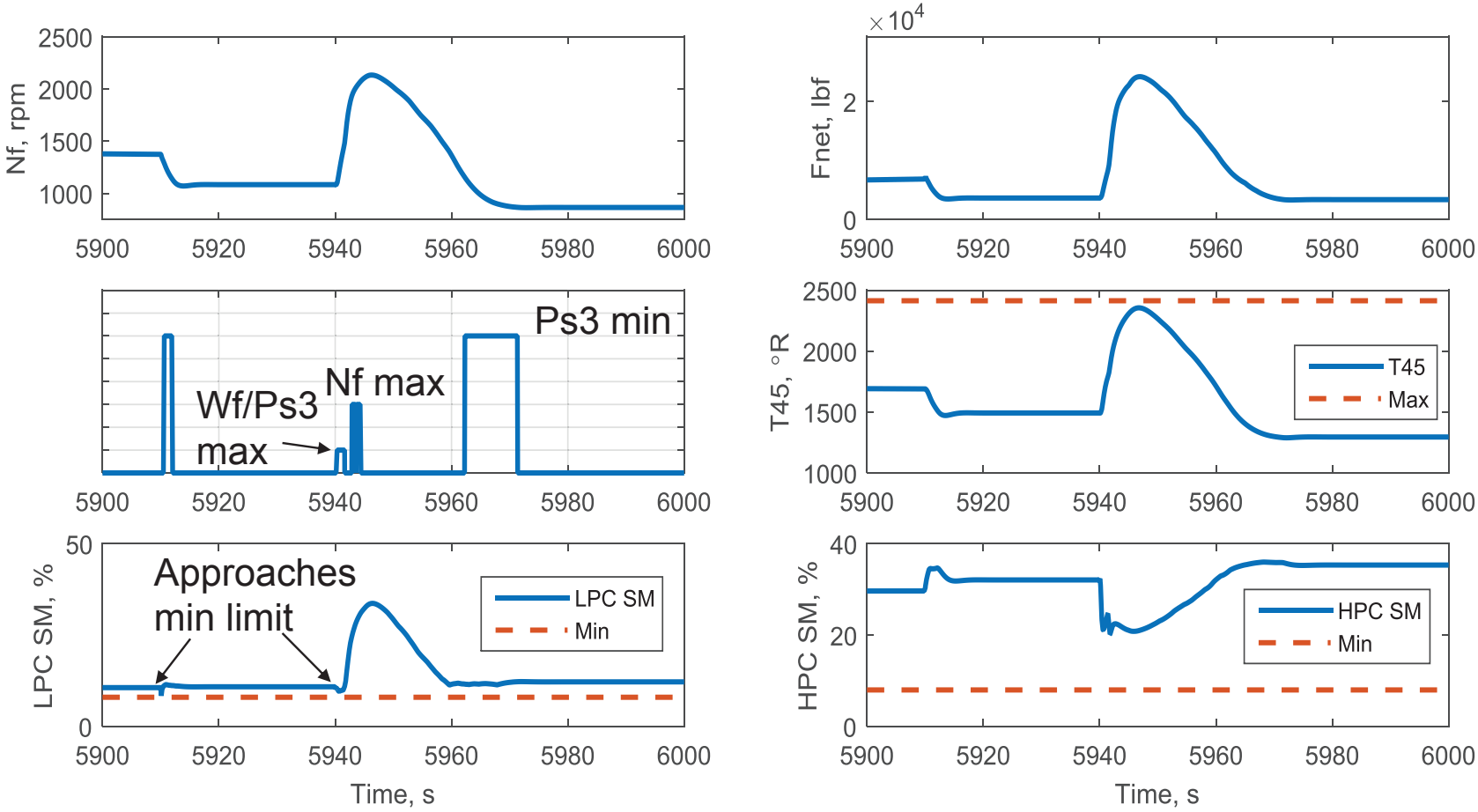
For the validation profile, all parameters remain within acceptable ranges and the engine performs as expected

Model Validation, takeoff and climb



During acceleration and climb to altitude the control regulators act to maintain stall margin and maximum T45 limit

Model Validation, approach and landing



During approach and landing the control regulators act to maintain stall margin, maximum Nf limit and minimum Ps3 limit

Summary

- A simulation of a next generation engine has been presented
  - Advanced Geared Turbofan 30,000lbf (AGTF30)
    - Ultrahigh bypass, small engine core, VAFN design
    - Full envelope dynamic control system
    - Built with the Toolbox for the Modeling and Analysis of Thermodynamic systems (T-MATS), https://github.com/nasa/T-MATS/releases
    - Planned to be made publicly available
- Control system design described
  - Fuel control based on classical architecture
  - Variable geometries scheduled
- AGTF30 simulation meets all requirements
  - Simulation provides a realistic and dynamic platform for research into advanced geared turbofan technologies.

NASA/CP—2018-219785

Acknowledgments

Funding for this work was provided by the Transformative Aeronautics Concepts Program (TACP)/Transformational Tools and Technologies (TTT) project and the Advanced Air Vehicles Program (AAVP)/Advanced Air Transport Technology (AATT) project.

Demo

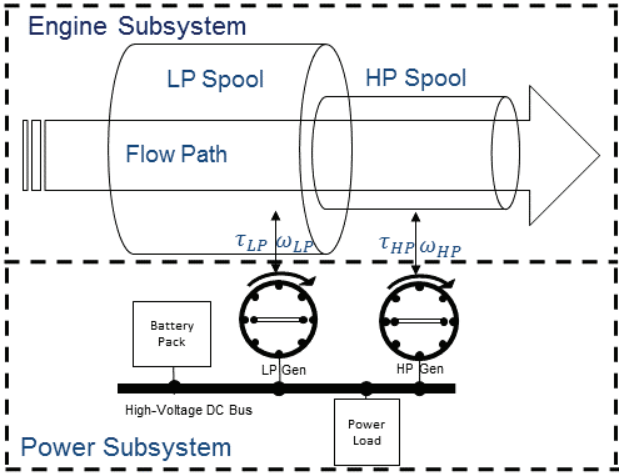
NASA/CP—2018-219785

Toolbox for the Modeling and Analysis of Thermodynamic Systems (T-MATS) Users' Workshop

Applications

MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Modeling an Aircraft Propulsion Subsystem for Developing Coordinating Controllers in a More Electric Aircraft Using T-MATS



William Dunham
Department of Aerospace Engineering
University of Michigan

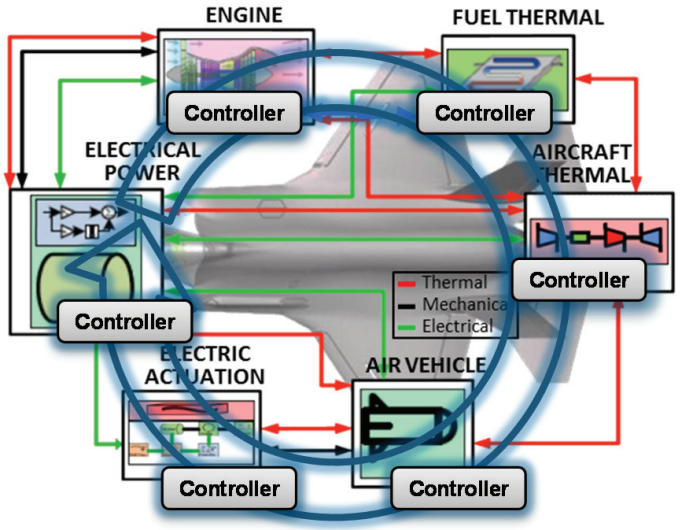
Acknowledgements: Ilya Kolmanovsky, Anouck Girard, Brandon Hencey, and Jinwoo Seok
This research was supported by the US Air Force Research Laboratory

Motivation

Traditionally, aircraft subsystems are controlled separately

Increased power loads from electro-mechanical actuators and advanced avionics will impact other subsystems

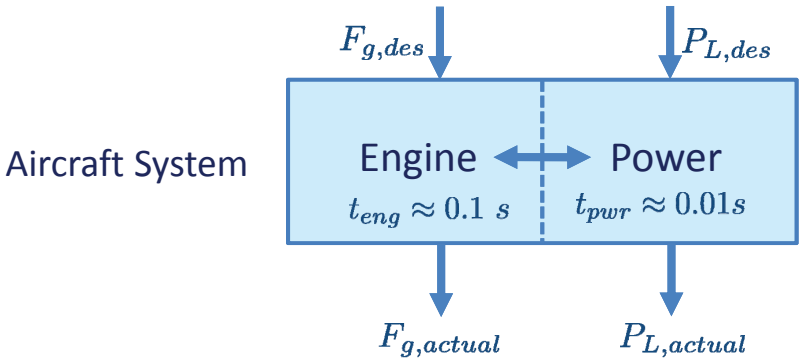
Each subsystem has local safety and specification constraints that must be met while following upper-level commands.



Approach

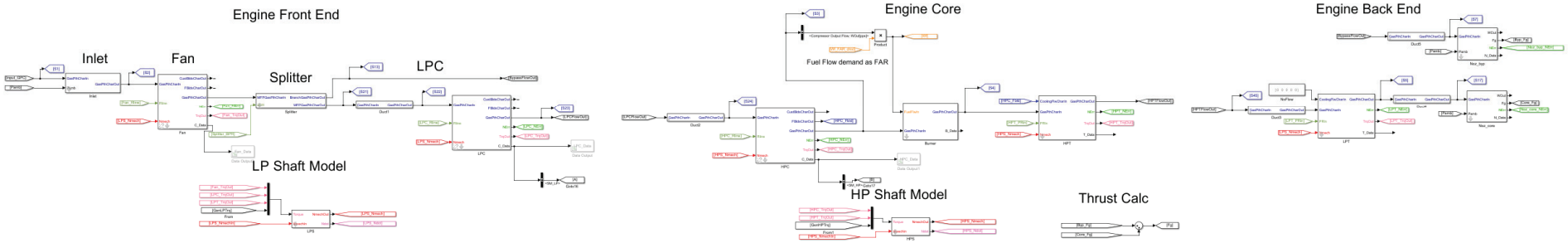
Separately controlled subsystems which communicate to improve tracking and constraint enforcement

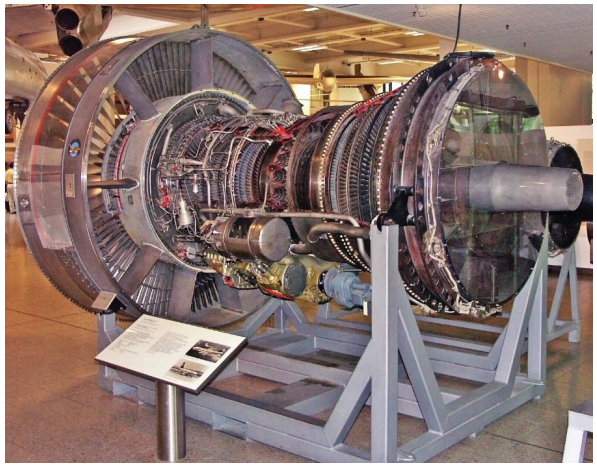
Require models of an aircraft engine and power subsystem including interactions between them



MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Engine Model – Modified JT9D Example





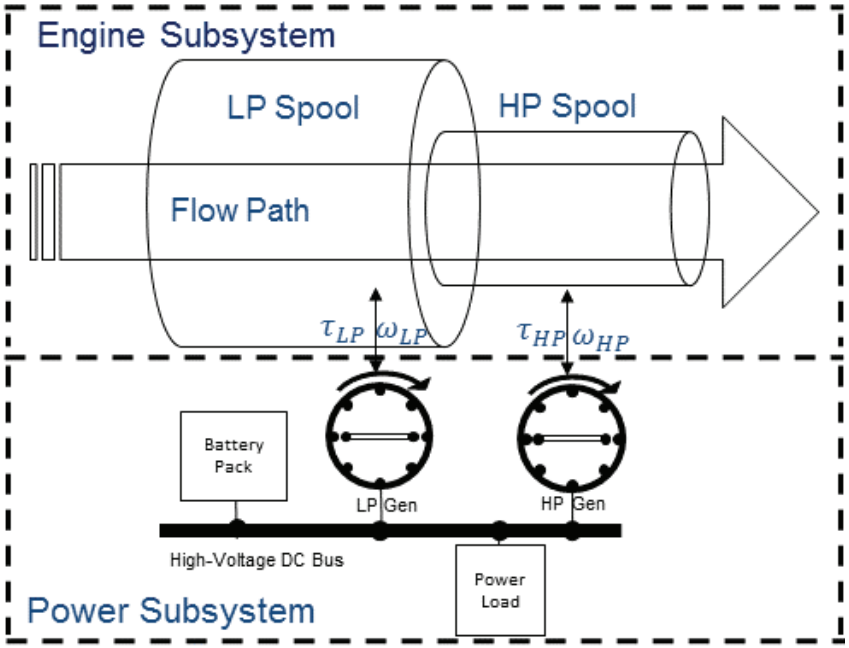
T-MATS is used as it:

- Models the thermodynamics at the desired level of detail and computational speed
- Works in the Simulink graphical environment (a big plus for a controls engineer)
- Is packaged with a working turbine example which can be easily followed and modified

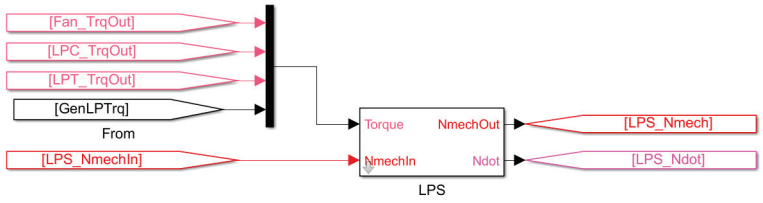
Interactions w/Generators

The models are mechanically coupled through the spool shafts

The engine drives the generator shafts which feedbacks a torque



LP Shaft Model



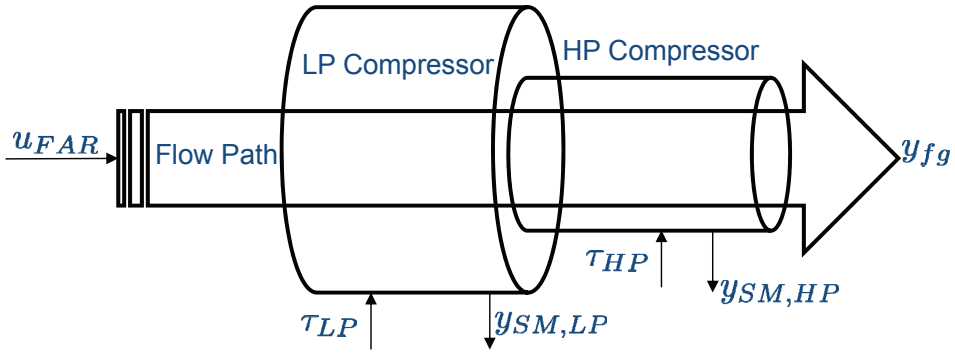
Engine From a Controls Perspective

Controlled input (u_e):

u_{FAR} : Fuel-to-air ratio entering the engine flow path

Outputs (y_e):

$y_{SM,LP}$: Surge Margin of the LP Compressor
$y_{SM,HP}$: Surge Margin of the HP Compressor
y_{fg} : Thrust force out from the engine



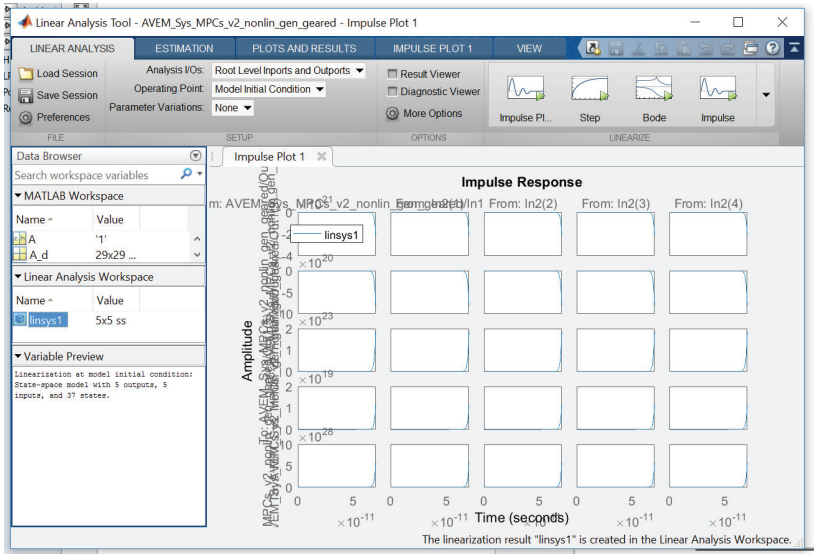
Interaction variables:

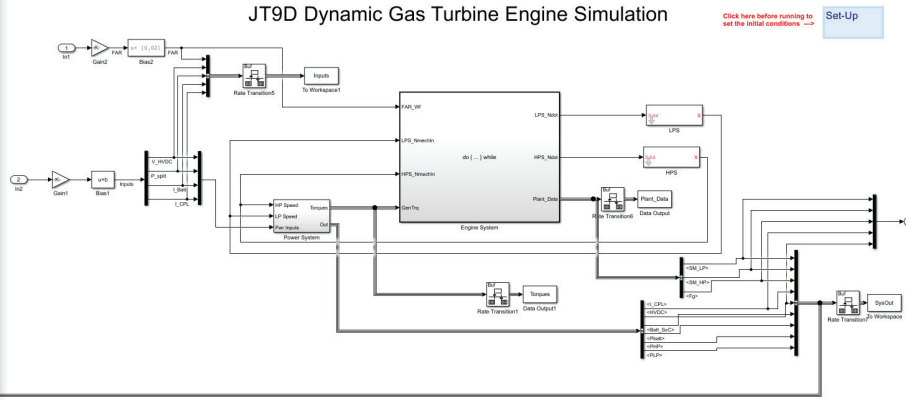
$\tau_{HP/LP}$: Electrical generator torques

MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Simplified Model

In order to apply model predictive control, a linear model is needed. We've found that the Linear Analysis toolbox works well in identifying linear models without much difficulty. Additionally, data can be manually collected for the System ID Matlab application.



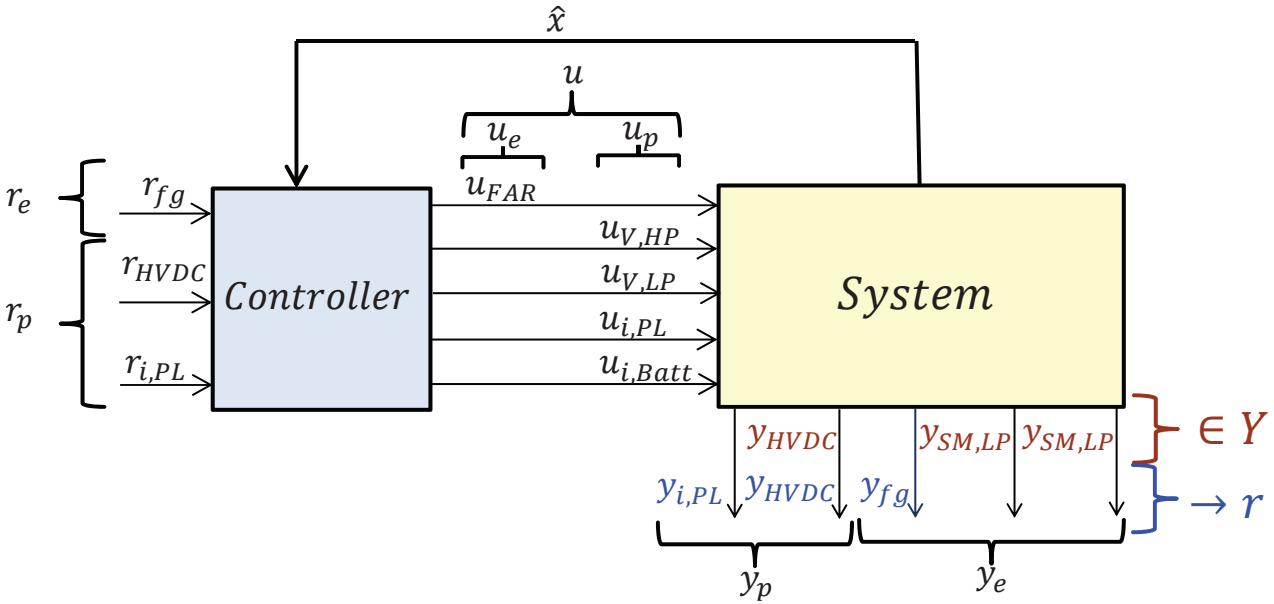


MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Control problem

Accurately track set-points r_e, r_p in order to provide mission critical power and thrust while satisfying constraints.

Assume (current work) all states needed for control are measured or estimated.



Prediction model

A linearized model, converted to discrete-time ($T_s = 0.01$ sec)

$$\delta x^{k+1} = A\delta x^k + B\delta u^k,$$
$$y^k = Cx^k + Du^k + y^{\text{nom}}.$$

The model can also be split as

$$\begin{bmatrix} \delta x_p^{k+1} \\ \delta x_e^{k+1} \end{bmatrix} = \begin{bmatrix} A_{pp} & A_{pe} \\ A_{ep} & A_{ee} \end{bmatrix} \begin{bmatrix} \delta x_p^k \\ \delta x_e^k \end{bmatrix} + \begin{bmatrix} B_{pp} & B_{pe} \\ B_{ep} & B_{ee} \end{bmatrix} \begin{bmatrix} \delta u_p^k \\ \delta u_e^k \end{bmatrix},$$

$$\begin{bmatrix} \delta y_p^k \\ \delta y_e^k \end{bmatrix} = \begin{bmatrix} C_p & 0 \\ 0 & C_e \end{bmatrix} \begin{bmatrix} \delta x_p^k \\ \delta x_e^k \end{bmatrix} + \begin{bmatrix} D_p & 0 \\ 0 & D_e \end{bmatrix} \begin{bmatrix} \delta u_p^k \\ \delta u_e^k \end{bmatrix}.$$

Rate-based model

Let $e^k = y^k - r^k, \quad \Delta x^k = x^{k+1} - x^k, \quad \Delta u^k = u^{k+1} - u^k$

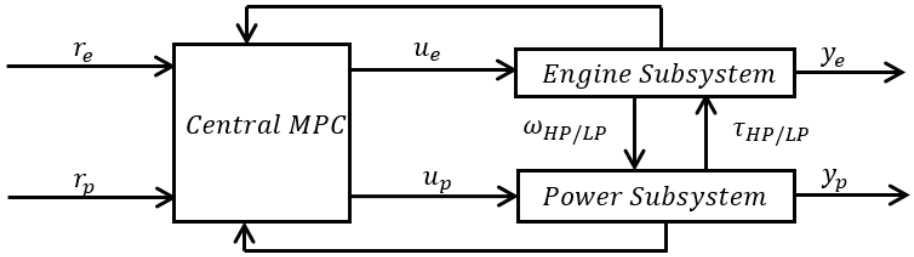
$$x_{ext}^k = [(\Delta x^k)^T \ (e^k)^T \ (y^k)^T \ (u^k)^T]^T$$

Then

$$\begin{bmatrix} \Delta x^{k+1} \\ e^{k+1} \\ y^{k+1} \\ u^{k+1} \end{bmatrix} = \begin{bmatrix} A & 0 & 0 & 0 \\ C & I & 0 & 0 \\ C & 0 & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix} \begin{bmatrix} \Delta x^k \\ e^k \\ y^k \\ u^k \end{bmatrix} + \begin{bmatrix} B \\ D \\ D \\ I \end{bmatrix} \Delta u^k,$$

$$x_{ext}^{k+1} = A_{ext} x_{ext}^k + B_{ext} \Delta u^k.$$

Centralized rate-based MPC (Cent)



$$\Delta U = \begin{bmatrix} \Delta u^0 \\ \Delta u^1 \\ \vdots \\ \Delta u^{N_h-1} \end{bmatrix}$$

$$\mathcal{J}(\Delta U) = \sum_{k=0}^{N_h-1} \left(\|x_{ext}^k\|_Q^2 + \|\Delta u^k\|_R^2 \right) + \|x_{ext}^{N_h}\|_P^2 \to \min_{\Delta U}$$

subject to

$$x_{ext}^{k+1} = A_{ext} x_{ext}^k + B_{ext} \Delta u^k, \quad k = 0, \cdots, N_h - 1$$

$$x_{ext}^0 = x_{ext}(t)$$

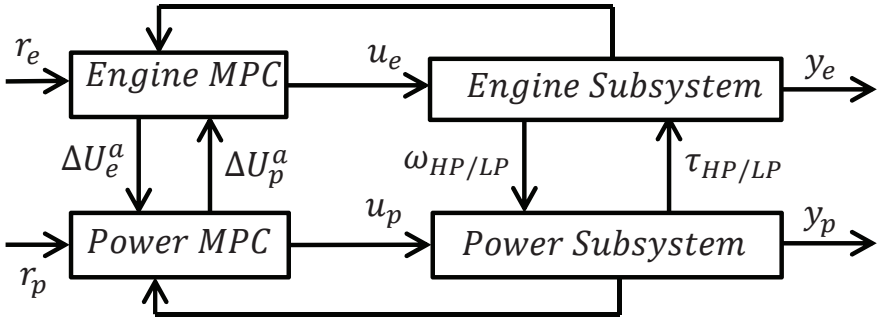
affine constraints $\Delta U \in \Delta \mathbb{U}(x_{ext}(t))$

MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Distributed, Cooperative MPC (CD)

Controllers exchange information during optimization iterations

Each controller has access to all subsystem costs and states



$$\Delta U_e = \begin{bmatrix} \Delta u_e^0 \\ \Delta u_e^1 \\ \vdots \\ \Delta u_e^{N_h-1} \end{bmatrix} \qquad \Delta U_p = \begin{bmatrix} \Delta u_p^0 \\ \Delta u_p^1 \\ \vdots \\ \Delta u_p^{N_h-1} \end{bmatrix}$$

$$\mathcal{J}(\Delta U_e, \Delta U_p) = \mathcal{J}_e(\Delta U_e, \Delta U_p) + \mathcal{J}_p(\Delta U_e, \Delta U_p)$$

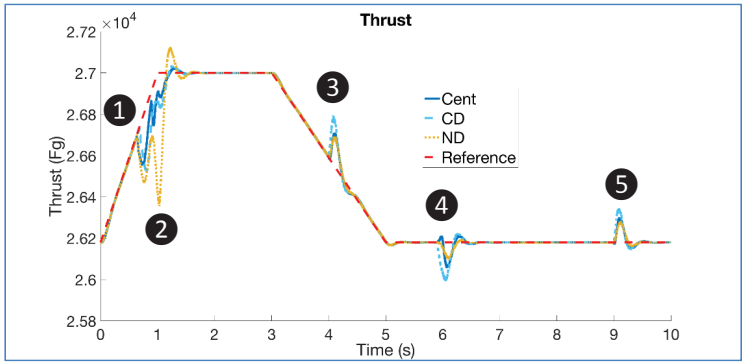
$$\mathcal{J}(\Delta U_e, \Delta U_p^a) \to \min_{\Delta U_e} \quad \Rightarrow \Delta U_e^{a+1}$$

$$\mathcal{J}(\Delta U_e^a, \Delta U_p) \to \min_{\Delta U_p} \quad \Rightarrow \Delta U_p^{a+1}$$

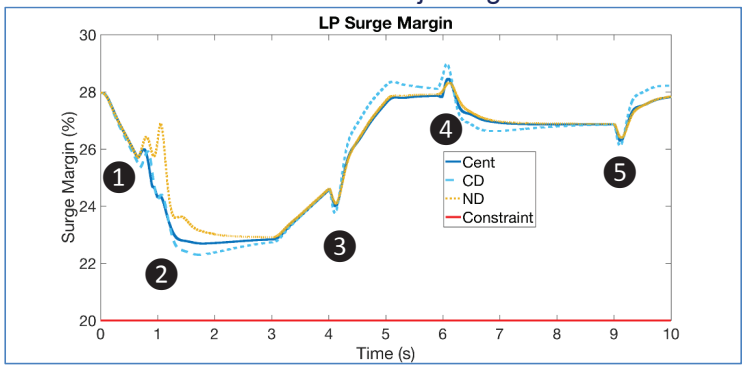
$$\Rightarrow \Delta U_e^*(t),\ \Delta U_p^*(t)$$

$$a := a + 1$$

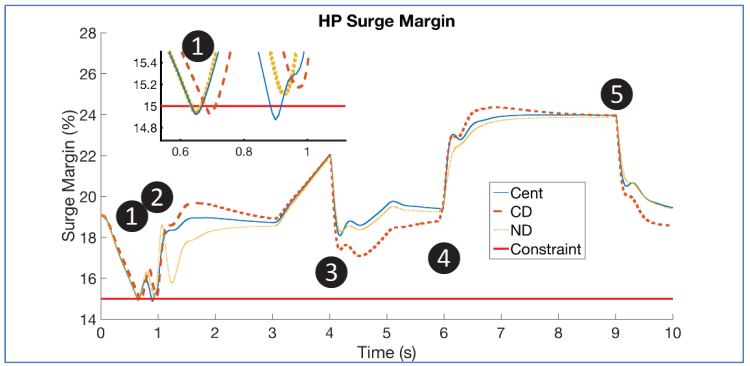
Engine subsystem results



Total thrust from the jet engine.



Low Pressure Compressor Surge Margin.



High Pressure Compressor Surge Margin.

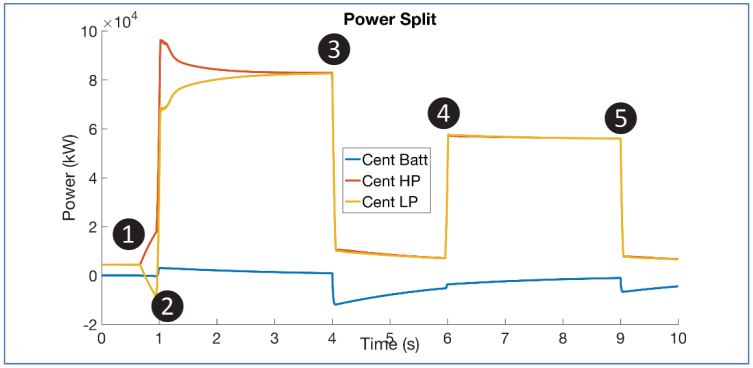
❶ At roughly t=0.6 sec, the HP surge margin constraint is violated by all controllers. Thrust tracking performance is affected, however Cent and CD are assisted by the power subsystem. The insert shows that the Cent controller actually violates the constraint twice but all three remain close.

❷ At t=1 sec, the power load is greatly increased. The new torques push the HP surge margin away from the constraint and the ND controller regains thrust tracking.

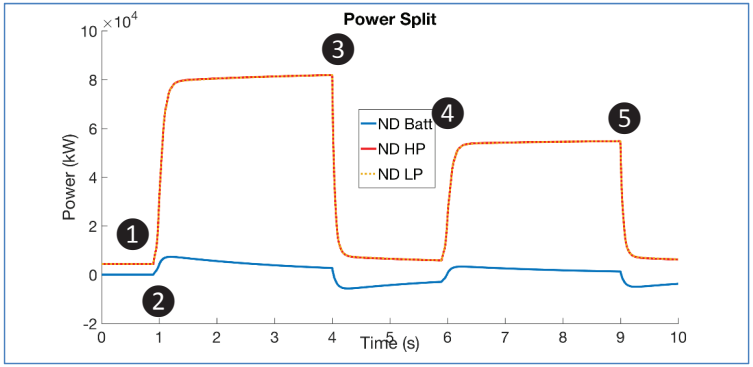
❸ At t=4 sec, the power load is dropped.

❹, ❺ A second, smaller power load comes on at t=6 sec and ends at t=9 sec.

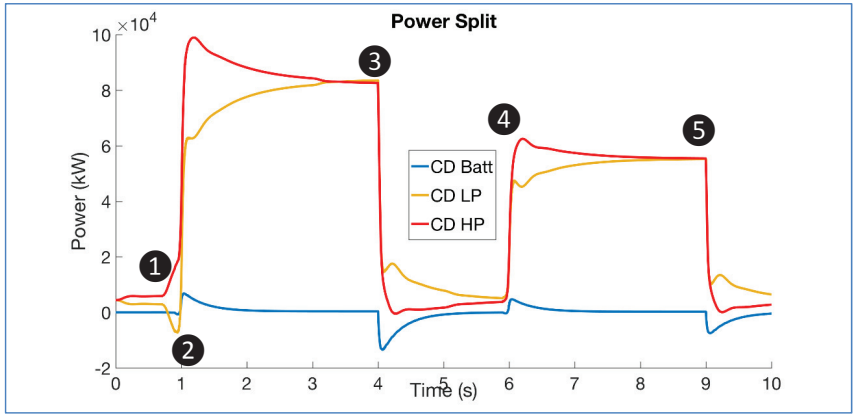
Power subsystem results – power splits



Battery, HP, and LP Generator power loads for Central case.



Battery, HP, and LP Generator power loads for ND case.



Battery, HP, and LP Generator power loads for CD case.

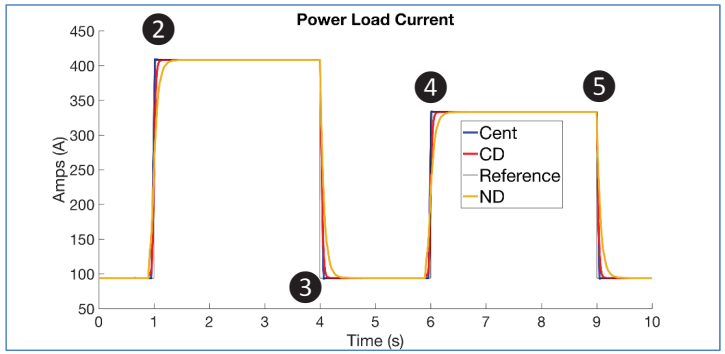
❶ At roughly t=0.6 sec, the Cent and CD controllers split the power loads unevenly over the two generators in response to the HP surge margin constraint in the engine subsystem.

❷ At t=1 sec, the power load is greatly increased. The batteries are used in transient for HVDC voltage stability while the generators supply the bulk of the load.

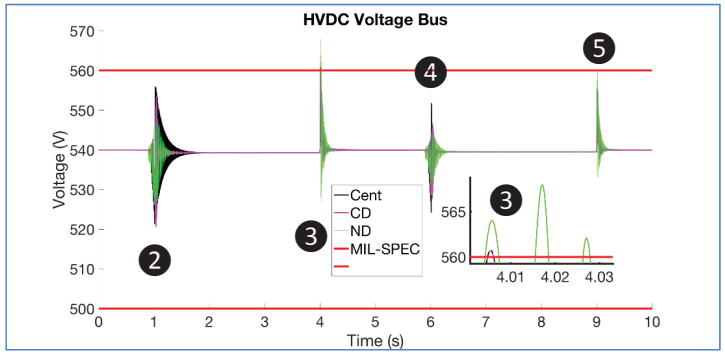
❸ At t=4 sec, the power load is dropped.

❹, ❺ A second, smaller power load comes on at t=6 sec and ends at t=9 sec.

Power subsystem - outputs



Power load current.



HVDC Voltage.

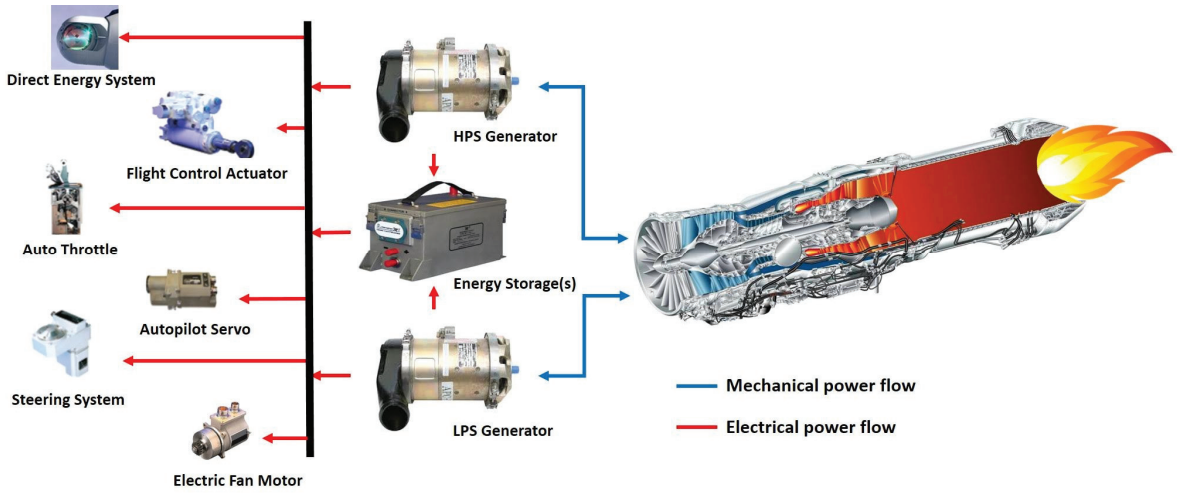
❷ At t=1 sec, the power load is greatly increased. The batteries are used in transient for HVDC voltage stability while the generators supply the bulk of the load. The ND controller lags behind the other two in reaching the reference power load current.

❸ At t=4 sec, the power load is dropped. The Cent controller has an acceptable voltage violation, according to MIL-SPEC. The ND controller violates the constraint repeatedly in rapid succession, which is not acceptable.

❹, ❺ A second, smaller power load comes on at t=6 sec and ends at t=9 sec.

Concluding Remarks

T-MATS is a powerful and robust tool for control design. It's robustness and speed enables the testing of interesting system configurations and control scenarios.



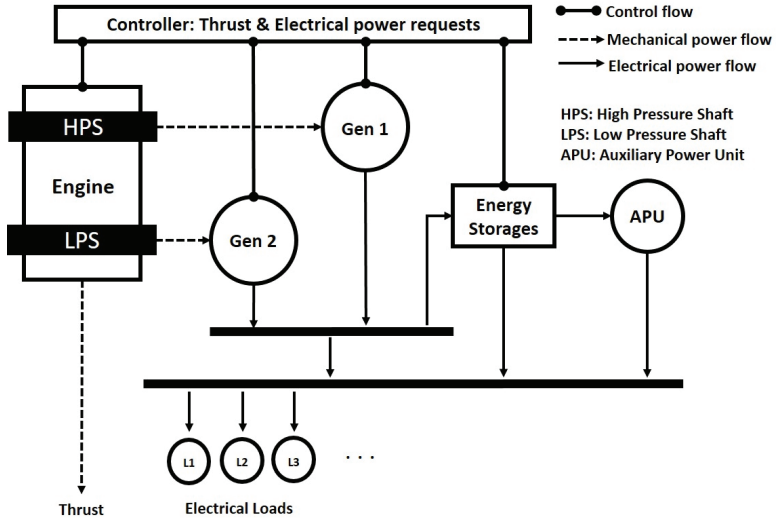
Publications from this Model

Seok, J., Kolmanovsky, I., and Girard, A., "Coordinated Model Predictive Control of aircraft gas turbine engine and power system," *AIAA Journal of Guidance, Control and Dynamics*, 2017.

Dunham, W. Hencey, B. Kolmanovsky, I., and Girard, A., 2017. "Predictive propulsion and power control for large transient power loads in a more electric aircraft". *American Controls Conference (ACC)*.

Dunham, W. Hencey, B., Girard, A., and Kolmanovsky, I., 2017. "Distributed MPC via ADMM for Coordination and Control of More Electric Aircraft Sybsystems", *DSCC*, under review

….and more to come!




MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN
Questions, comments, or suggestions??

Applications of T-MATS to Hardware-in-the-Loop Simulation Modeling

2017 TMATS Workshop

George Thomas

N&R Engineering

Intelligent Control & Autonomy Branch

August 21, 2017

Outline

- Introduction
  - Distributed Engine Control (DEC)
  - Distributed Engine Control System Simulator (DECSS)
- HIL Test Design
- Prototype AGTF30 Distributed Control Network
- Conclusions

Introduction

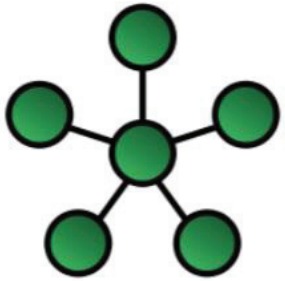
- Objectives:
  - Develop infrastructure for HIL test of distributed engine control (DEC) technologies
  - Be able to test
    - Variety of engine plant simulations in system with hardware
    - Advanced control techniques/logic
    - Prototype hardware (smart node, communications devices)
    - Combinations of hardware and software
  - HIL test infrastructure allows improving DEC TRL and evaluating system benefits
- Plan:
  - Leverage TMATS to model engines with DEC devices/concepts
  - Conduct HIL tests for DEC research
    - Use existing NASA GRC lab capabilities & add more as necessary

NASA/CP—2018-219785

Introduction – DEC

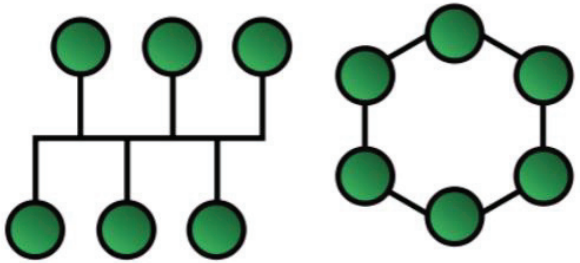
- **Distribution** of previously **centralized** control elements
  - Different architecture, performs **same functions** as centralized
  - However, **indirect benefits** due to reduced system constraints
    - Fuel burn (nacelle diameter, cabling weight)
    - Cost (maintenance, design, life cycle)
  - Also **enables future capability**
    - Faster local control with slower supervisory control
    - Greater computing resources

Centralized



vs

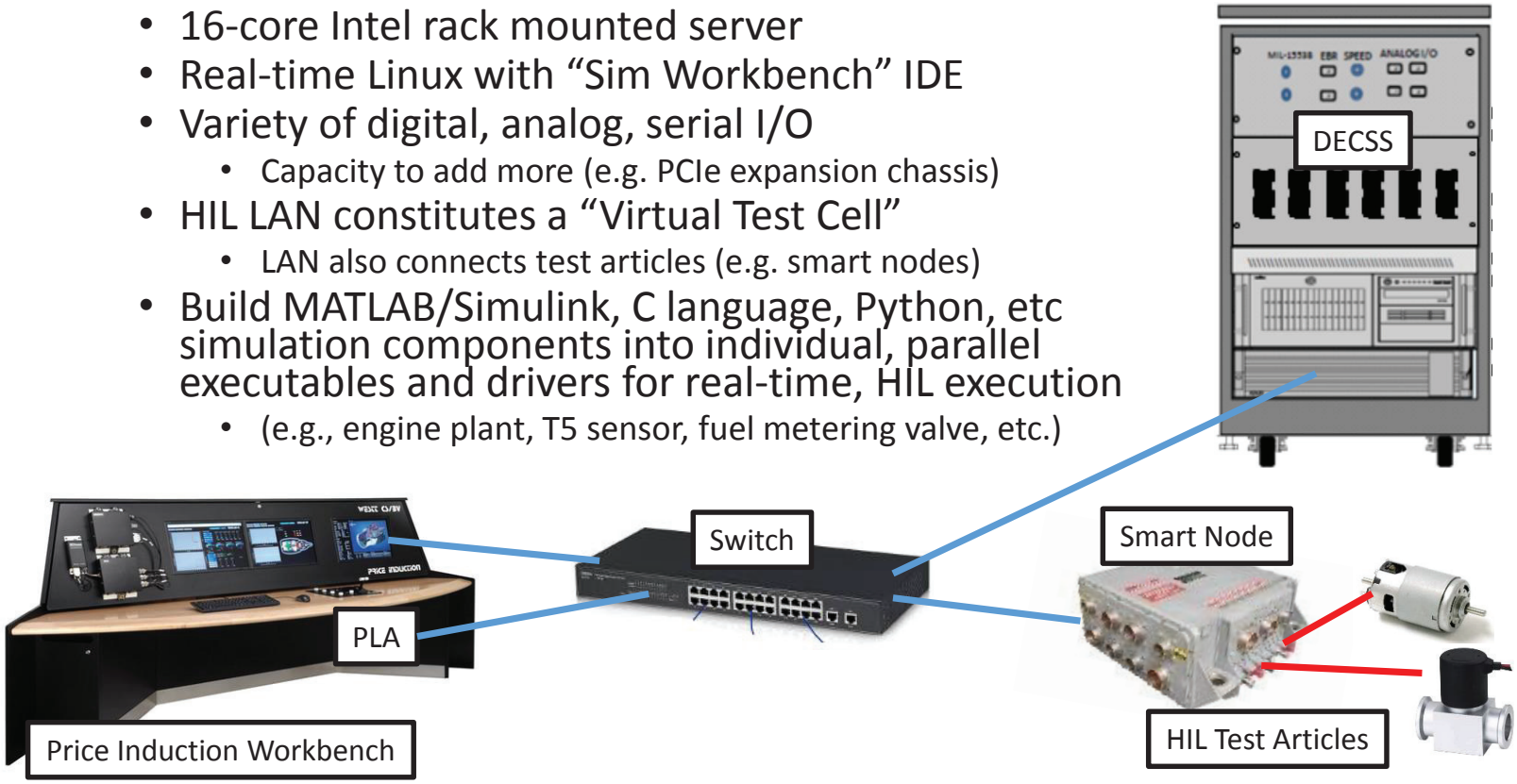
Distributed



NASA/CP—2018-219785

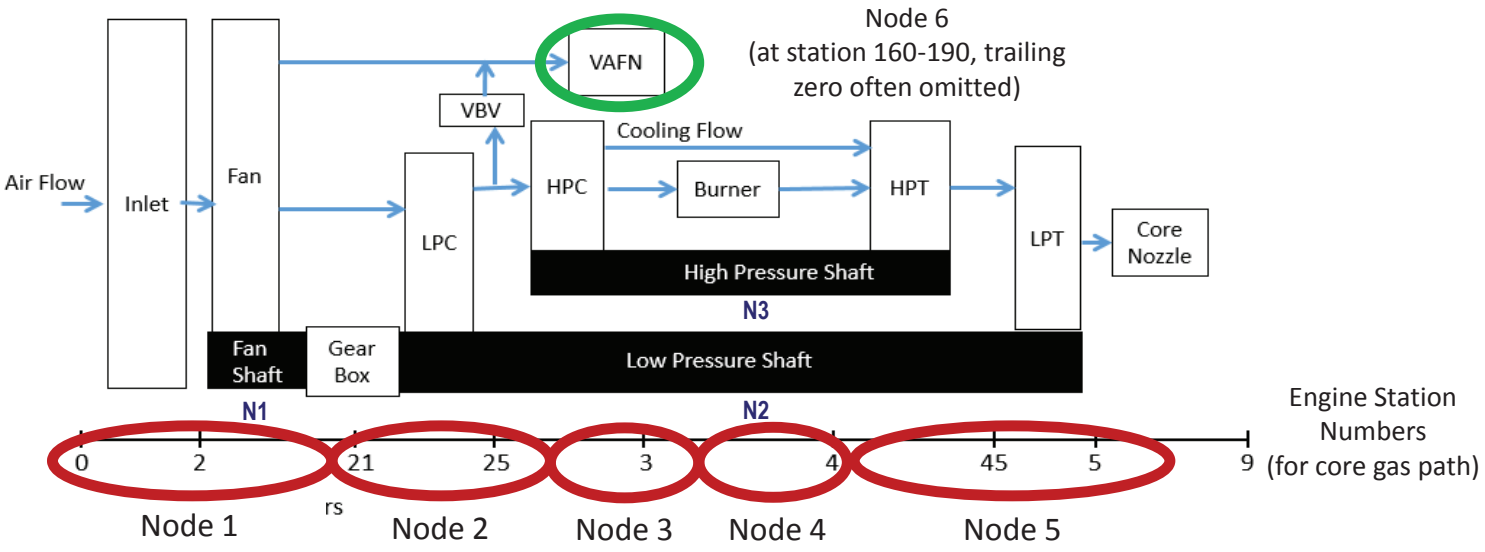
Introduction – DECSS

- Distributed Engine Control System Simulator
  - 16-core Intel rack mounted server
  - Real-time Linux with “Sim Workbench” IDE
  - Variety of digital, analog, serial I/O
    - Capacity to add more (e.g. PCIe expansion chassis)
  - HIL LAN constitutes a “Virtual Test Cell”
    - LAN also connects test articles (e.g. smart nodes)
  - Build MATLAB/Simulink, C language, Python, etc simulation components into individual, parallel executables and drivers for real-time, HIL execution
    - (e.g., engine plant, T5 sensor, fuel metering valve, etc.)



Prototype AGTF30 DEC Network

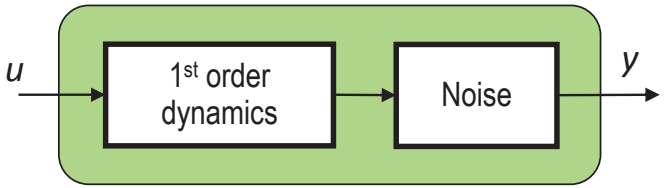
- AGTF30: Advanced geared turbofan concept engine in TMATS
  - Concept/demo DEC architecture built around this engine
  - Sensing and actuation responsibilities grouped by station location
  - Groups shown with circles, (red are core locations, green are bypass)
  - Each group is assigned a particular smart node



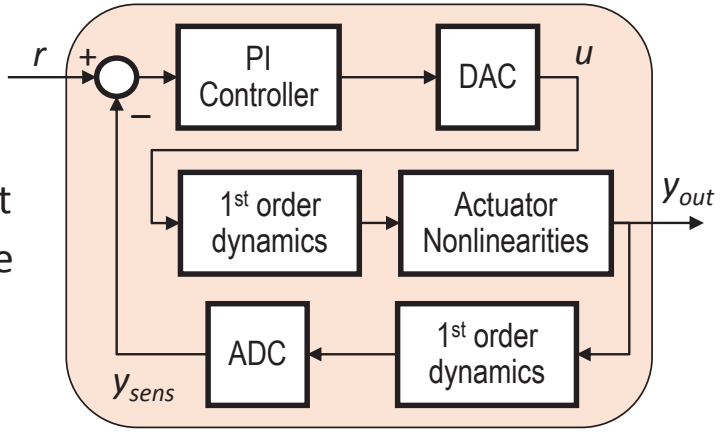
Prototype AGTF30 DEC Network

- **Sensor simulation logic**
  - 1st order linear dynamics
  - Noise
- **Actuator simulation logic**
  - Local PI loop containing:
    - PI control signal DAC
      - Quantization, reconstruction filter
    - 1st order actuator dynamics with noise
    - Actuator nonlinearities (final response)
      - Backlash (deadband) and slew rate limit
    - 1st order feedback sensor dynamics + noise
      - Feedback sensor ADC
    - Quantization, anti-aliasing filter
- Made these blocks consistent with TMATS library block format

Sensor Simulation Model

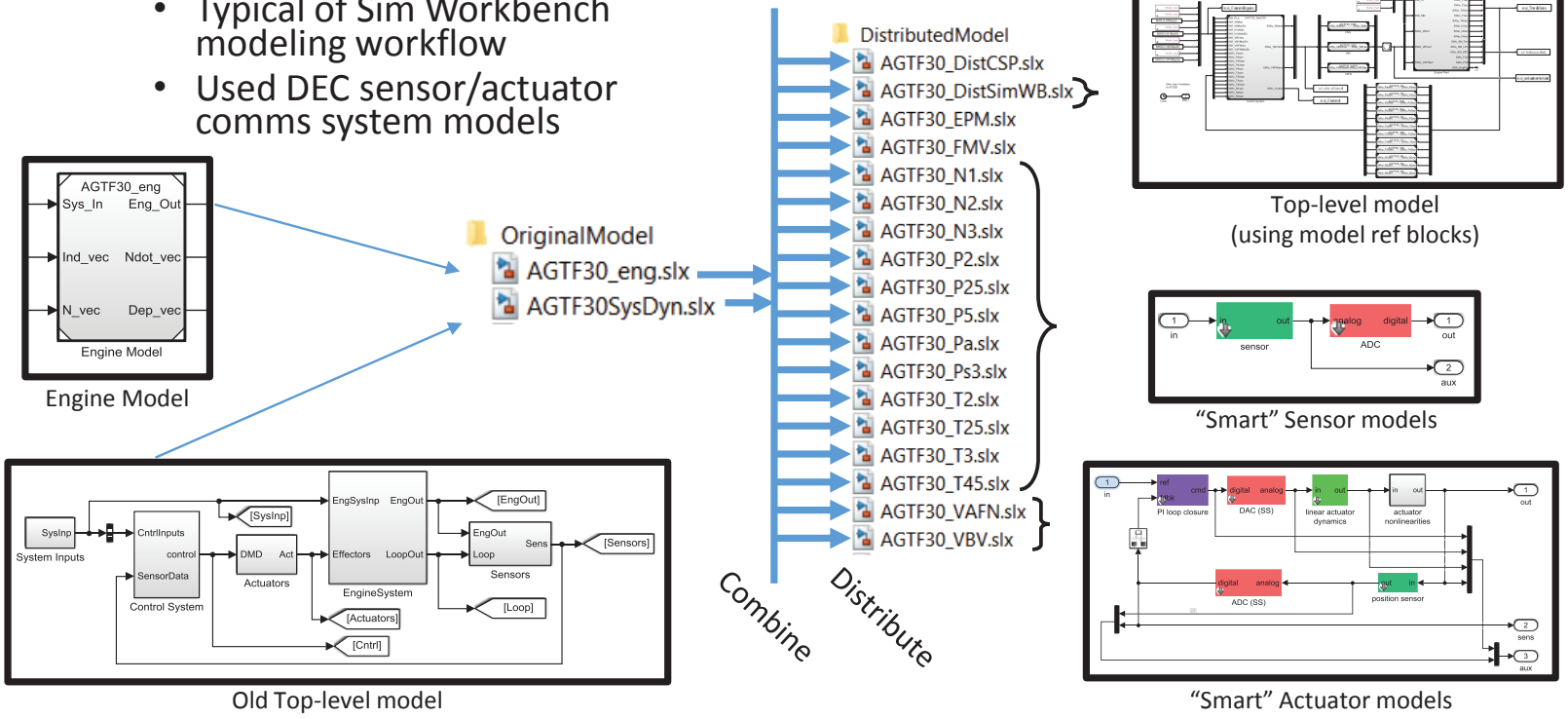


Actuator Simulation Model



Prototype AGTF30 DEC Network

- Process to create distributed AGTF30 model for HIL
  - Took AGTF30 model and distributed its system components into individual model reference blocks that live in a "top-level" model
  - Typical of Sim Workbench modeling workflow
  - Used DEC sensor/actuator comms system models



Prototype AGTF30 DEC Network

- Build, deploy, and run on DECSS using Sim WB API script
  - Takes in “top-level model” and parses it to find model ref blocks to build
  - Creates real-time database (shared memory) from parsed models
  - Copies all of these models to TMATS_Library/MEX folder and builds there
    - Solves Simulink Coder “can’t find dependency” issues
  - Deletes copies of model files to restore original MEX folder

Setup

```
%
% Due to limitations in Sim Workbench's automatic makefile generation code,
% we want to build the simulink models in the same folder as the
% S-Functions and their C code. So let's copy the simulink files over to
% /TMATS/TMATS_Library/MEX, build the executable there, then delete all of
% the new files we made in there to keep things clean, and CD back when
% we're finally done. Sounds good?
%
% Also, make sure you have the MMS structure in memory that contains the
% simulation variables that won't change after the build.
% (initial conditions, controller parameters, etc.)
% i.e.: run ">> setup_everything" first.
%

% Create backups of MATLAB path and MATLAB working directory.
originalPath = path;
originalDirectory = cd;

% CD to the folder this file is in (we're assuming that this function is in
% "AGTF30 root", and that the TMATS mex folder can be found at
% "AGTF30 root"/TMATS/TMATS_Library/MEX
[directory] = fileparts(mfilename('fullpath'));
cd(directory);

% Get a list of what files are currently in the TMATS mex folder so we can
% delete any files that are not in the list when we leave.
temp = dir('TMATS/TMATS_Library/MEX');
mexFolderFileWhitelist = {temp.name};

% Get list of all of our DEC AGTF30 simulink models so we can copy them
% over.
temp = dir('DistributedModel/*.slx');
filenames = {temp.name};

% Copy them
for i = 1 : length(filenames)
   copyfile(['DistributedModel/',filenames{i}],['TMATS/TMATS_Library/MEX/',filenames{i}])
end

% CD to the build folder and add build dependencies to path.
cd('TMATS/TMATS_Library/MEX/');
addpath(genpath('..'));
```

Build each sub-model

```
% For each referenced model in the top-level model
for iModel=1:length(modelList)
    % Populate RTWStruct
    % Use model created in Getting Started tutorial video
    rtobj.RTWStruct.modelname = modelList{iModel}; % Name of the model
    rtobj.RTWStruct.rtdb      = RTDBname;    % Name of the RTDB
    % System Target File is required for generating code
    rtobj.RTWStruct.systemtargetfile =  'TLCfile';
    % SWprefix option take values 0, 1, or 2
    % 0 is all source sinks, 1 is reg expressions, and 2 is both
    rtobj.RTWStruct.SWprefix = 1 ;
    rtobj.RTWStruct.sourceRegExp=sourceRegExp;
    rtobj.RTWStruct.sinkregexp=sinkRegExp;

    % Create RTDB for the my_sldemo_auspn model
    rtobj.createRTDB;

    % Set initial conditions for "SWio" RTDB variables by adding them to
    % the signal.db file.
    signalDBFilestring = fileread('signal.db');
    signalDBFilestring = setRTDBICs(signalDBFilestring);
    filePointer = fopen('signal.db','w');
    fprintf(filePointer, signalDBFilestring);
    fclose(filePointer);

    % Upload the rtdb with added initial conditions.
    rtobj.uploadRTDB;
    % Get the name of the configuration RTDB
    rtobj.getConfigRTDBName;
    % Check model compliance with the configuration RTDB
    [matched_inport, matched_outport, unmatched_inport, unmatched_outport] ...
        = rtobj.checkCompliance;

    % Generate code, tranfer source code to Real-Time Host, and create
    % executable on the Real-Time Host.
    rtobj.rtwBuild;

    close_system(modelList{iModel});

end % end for loop on elements of modelList
```

Clean up

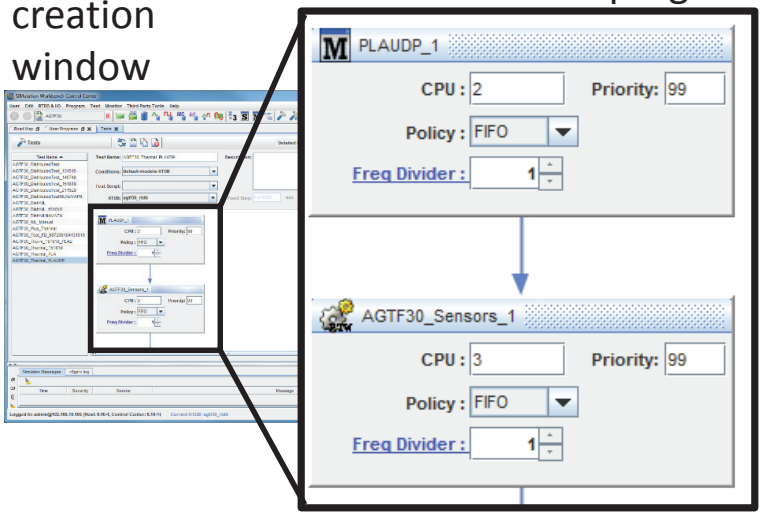
```
% You can disconnect MATLAB from the Simulation Workbench server
rtobj.disconnect

% Delete temporary build files and restore working directory and path.
% Start by getting build stuff off of our path and clearing mex and
% simulink stuff from memory.
clear mex; bdclose all;
path(originalPath);

% Get list of files that are now in the mex folder (aka the current folder)
% should be the originals plus some build stuff that we want to clean up.
temp = dir;
newFileList = {temp.name};

% For each file in the new list of files in the mex folder, iteratively
% check if it was in the original file list, and if it wasn't, delete it.
for i = 1 : length(newFileList)
    fileWasThereOriginally = false;
    for j = 1 : length(mexFolderFileWhitelist)
        if strcmp(newFileList{i},mexFolderFileWhitelist{j})
            fileWasThereOriginally = true;
        end
    end
    if ~fileWasThereOriginally
        if isdir(newFileList{i})
            rmdir(newFileList{i},'s')
        else
            delete(newFileList{i});
        end
    end
end
```

HIL Test Design

- Each model reference block compiled for real-time Linux
- All inport/outport variables having a given naming convention are added into the RTDB
- Take built models, RTDB, and create a “test” to be conducted

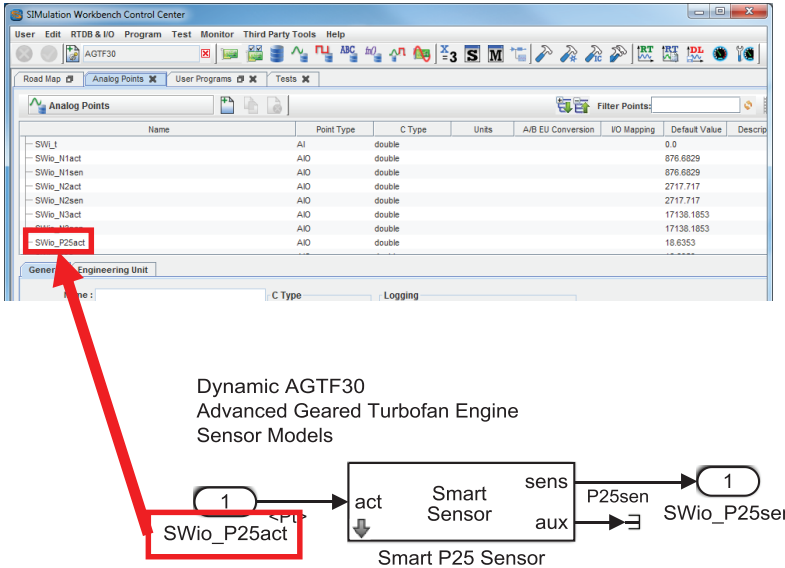
“Test” creation window

C-based interface program



Compiled Simulink/TMATS model

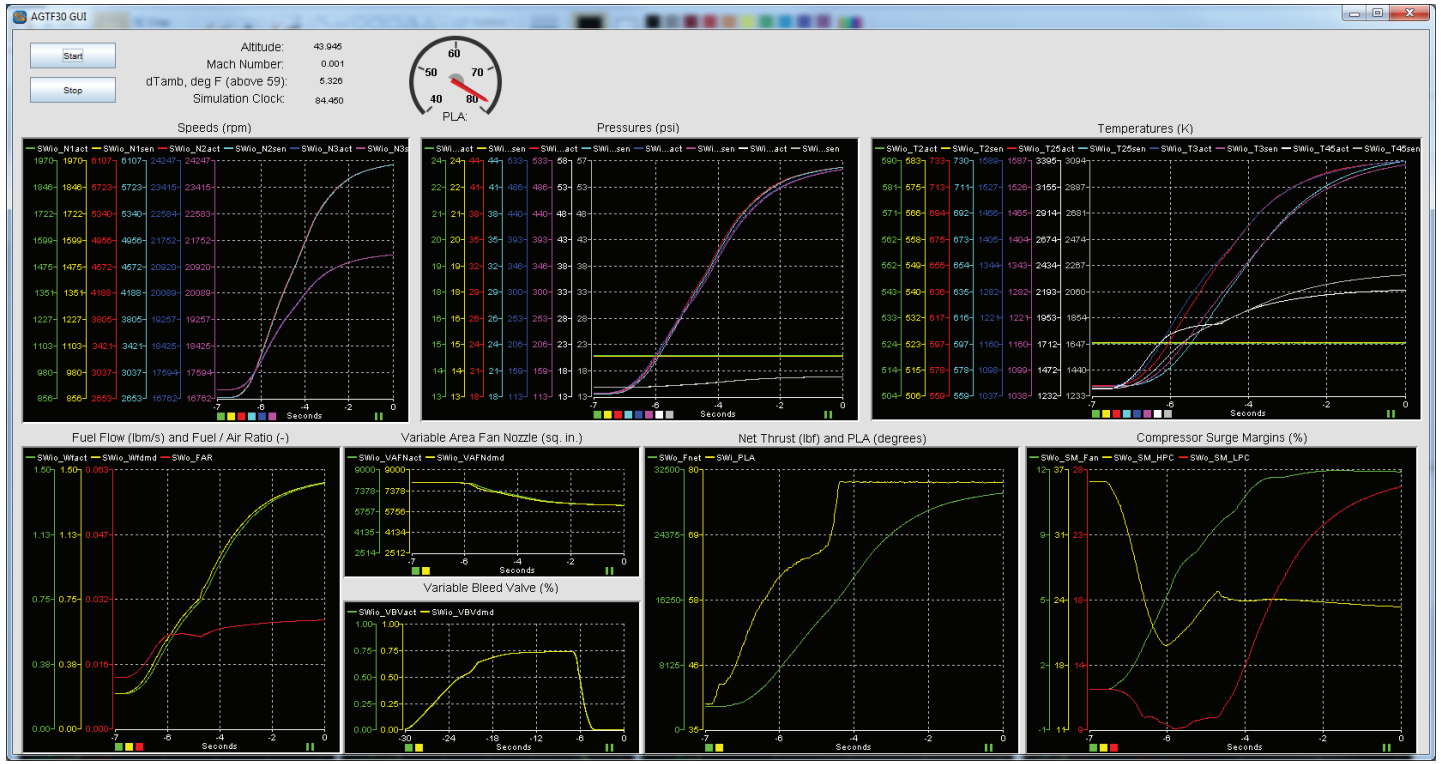
RTDB Variables, ICs, and hookups to IO



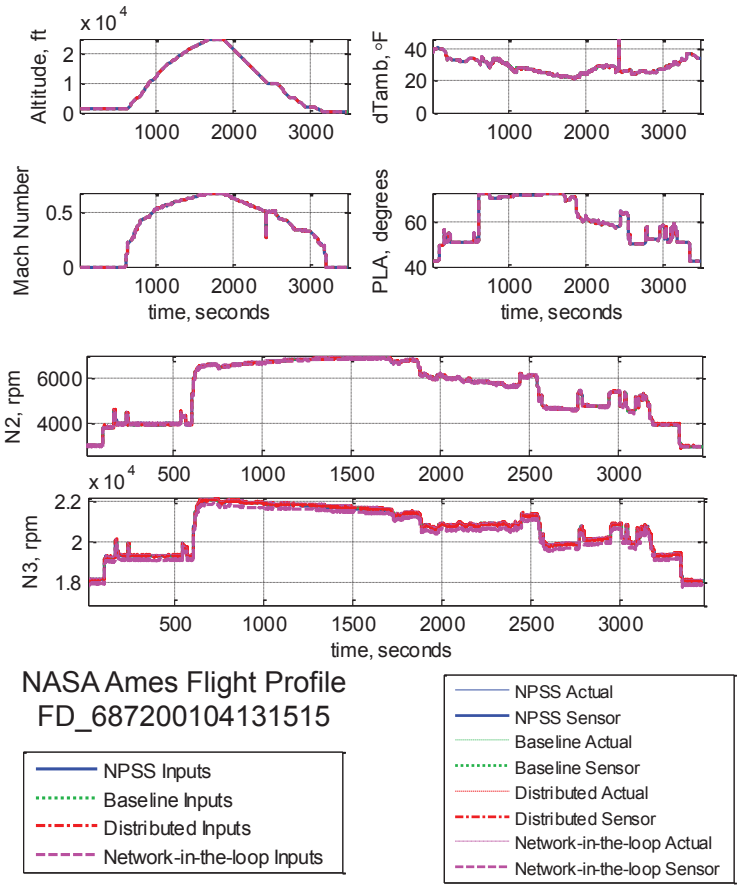
Model reference block name (RTDB var)

HIL Test Design

- Can construct a GUI and connect to hardware to do man-in-the-loop test with real-time TMATS engine



HIL Test Results



Results:

- **NPSS** (s-function) engine plant model with TMATS controller on Windows® platform
- **Baseline** TMATS AGTF30 engine model & controller, real time HIL platform
- **Distributed** TMATS AGTF30 engine plant model & controller with **simulated** DEC nodes and network on real time HIL platform
- **Network-in-the-Loop** TMATS AGTF30 engine plant model & controller with **physical** nodes and network on real time HIL platform
- Shows several capabilities
  - Can bring engine described in NPSS into TMATS-based HIL environment
  - Approach applies to any engine system
  - Can add DEC modeling fidelity to simulated control elements and compare with hardware
  - Appropriately designed DEC system (**Network-in-the-Loop**) successfully performs same function as centralized

Conclusions

- Demonstrated infrastructure for DEC system HIL test
- Can build TMATS model for real-time HIL simulation
- NASA GRC HIL test capabilities allow
  - Testing of
    - Different engine models in an environment with real hardware
    - Advanced control techniques/logic
    - Hardware prototypes of DEC devices
  - DEC system benefits and constraints to be investigated
  - New capabilities (e.g., active control) to be researched

References

1. Behbahani, A. R., “Achieving AFRL Universal FADEC Vision with Open Architecture Addressing Capability and Obsolescence for Military and Commercial Applications,” 42nd AIAA/ASME/SAE/ASEE Joint Propulsion Conference Exhibit, , No. AIAA 2006-4302, July 2006.
2. Culley, D. E., Thomas, R., and Saus, J., “Concepts for Distributed Engine Control,” Proceedings of the 43rd Joint Propulsion Conference and Exhibit, AIAA-2007-5709, Cincinnati, OH, July 2007.
3. Aretskin-Hariton, E. D., Zinnecker, A. M., Kratz, J. L., Culley, D. E., and Thomas, G. L., “Benchmarking model variants in development of a hardware-in-the-loop simulation system,” AIAA Science and Technology Forum and Exposition, AIAA-2016-1425, San Diego, CA, January 2016.
4. Zinnecker, A. M., Culley, D. E., and Aretskin-Hariton, E. D., “A modular approach to modeling hardware elements in distributed engine control systems,” AIAA Propulsion and Energy Forum and Exposition 2014: 50th AIAA/ASME/SAE/ASEE Joint Propulsion Conference, AIAA-2014-3530, Cleveland, OH, July 2014.
5. Aretskin-Hariton, E. D., Zinnecker, A. M., and Culley, D. E., “Extending the Capabilities of Closed-Loop Engine Simulation using LAN Communication,” AIAA Propulsion and Energy
6. Culley, D. E., Thomas, G. L., Aretskin-Hariton, E. D., “A Network Scheduling Model for Distributed Control Simulation,” 52nd AIAA/SAE/ASEE Joint Propulsion Conferencen, Salt Lake City, Utah, July 2016.
7. Thomas, G. L., Culley, D. E., and Brand, A. “The Application of Hardware in the Loop Testing for Distributed Engine Control.” 52nd AIAA/SAE/ASEE Joint Propulsion Conference (2016).
8. Scardelletti, M. C., Jordan, J. L., Meredith, R. D., et. al. “Demonstration of a Packaged Capacitive Pressure Sensor System Suitable for Jet Turbofan Engine Health Monitoring.” 2016 IEEE 66th Electronic Components and Technology Conference (ECTC) (2016)
9. Laurel, F., Usrey, M. W, and Watts, O, A. “Technical Opportunities for High Temperature Smart P3 Sensors and Electronics for Distributed Engine Control.” 52nd AIAA/SAE/ASEE Joint Propulsion Conference (2016).

Thank you!

Questions?

Target Code Generation From T-MATS Blocks

Jason Whitfield

MathWorks

Project Objectives

- Refactor the T-MATS block S-function code to enable code generation with MathWorks' *Embedded Coder* product
- Advantages:
  - Model Based Engine Control
  - Support for SIL and PIL simulation mode
  - Improved performance on HIL systems
- Demonstrate on embedded hardware

Refactoring the S-function Code

- Split S-function setup code and block calculation code into two files:
  - File containing S-function setup code.
    - Initializes Simulink block parameters.
    - Reads block input ports and mask parameters.
    - Passes inputs and parameters to calculation function.
    - Writes calculation results to block output ports.
  - File containing block calculation function.
    - Implements core block calculation.
    - Not reliant on S-function headers or code.

```
static void mdlInitializeSizes(SimStruct *S)
{
    int i;
    ssSetNumSFcnParams(S, NPARAMS);  /* Number of expected parameters */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        /* Return if number of expected != number of actual parameters */
        return;
    }

    for (i = 0; i < NPARAMS; i++)
        ssSetSFcnParamTunable(S, i, 0);

    ssSetNumContStates(S, 0);
    ssSetNumDiscStates(S, 0);

    if (!ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, 6);
    ssSetInputPortRequiredContiguous(S, 0, true);
    ssSetInputPortDirectFeedThrough(S, 0, 1);

    if (!ssSetNumOutputPorts(S, 1)) return;
    ssSetOutputPortWidth(S, 0, 6);

    ssSetNumSampleTimes(S, 1);
    ssSetNumRWork(S, 0);
    ssSetNumIWork(S, 0);
    ssSetNumPWork(S, 0);
    ssSetNumModes(S, 0);
    ssSetNumNonsampledZCs(S, 0);

}

static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
    ssSetModelReferenceSampleTimeDefaultInheritance(S);
}

#define MDL_START
#if defined(MDL_START)
static void mdlStart(SimStruct *S)
{
```

The Role of Target Language Compiler (TLC)

- Allows Simulink to generate setup code for target hardware.
- Uses same block calculation function as original S-function code.
- TMATS Thermo functions implemented entirely in TLC file.

```
%function Outputs(block, system) Output
    %%

    %assign out = LibBlockOutputSignal(0, "", "", 0)

    %assign in1 = LibBlockInputSignal(0, "", "", 0)
    %assign in2 = LibBlockInputSignal(0, "", "", 1)

    %<out> = t2hc(%<in1>, %<in2>);
    %%
%endfunction
```

Generated Code Comparison

Old T-MATS Code

```
Plant_GasTurbine_B.TmpSignalConversionAtCompressor[0] =
  Plant_GasTurbine_U.IS_In[0];
Plant_GasTurbine_B.TmpSignalConversionAtCompressor[1] =
  Plant_GasTurbine_B.AmbientEnvtoEngine[0];
Plant_GasTurbine_B.TmpSignalConversionAtCompressor[2] =
  Plant_GasTurbine_B.AmbientEnvtoEngine[1];
Plant_GasTurbine_B.TmpSignalConversionAtCompressor[3] =
  Plant_GasTurbine_B.AmbientEnvtoEngine[2];
Plant_GasTurbine_B.TmpSignalConversionAtCompressor[4] =
  Plant_GasTurbine_B.AmbientEnvtoEngine[3];
Plant_GasTurbine_B.TmpSignalConversionAtCompressor[5] =
  Plant_GasTurbine_U.Plant_In.OuterLoopPlantFeedback.Nmech;
Plant_GasTurbine_B.TmpSignalConversionAtCompressor[6] =
  Plant_GasTurbine_U.IS_In[1];
Plant_GasTurbine_B.TmpSignalConversionAtCompressor[7] = 1.0;
Plant_GasTurbine_B.TmpSignalConversionAtCompressor[8] = 10000.0;
Plant_GasTurbine_B.TmpSignalConversionAtCompressor[9] =
  Plant_GasTurbine_ConstB.s_C_Wc;
Plant_GasTurbine_B.TmpSignalConversionAtCompressor[10] =
  Plant_GasTurbine_ConstB.s_C_PR;
Plant_GasTurbine_B.TmpSignalConversionAtCompressor[11] =
  Plant_GasTurbine_ConstB.s_C_Eff;
```

New T-MATS Code

```
Plant_GasTurbine_B.TmpSignalConversionAtCompressor[0] =
  Plant_GasTurbine_U.IS_In[0];
Plant_GasTurbine_B.TmpSignalConversionAtCompressor[1] =
  Plant_GasTurbine_B.AmbientEnvtoEngine[0];
Plant_GasTurbine_B.TmpSignalConversionAtCompressor[2] =
  Plant_GasTurbine_B.AmbientEnvtoEngine[1];
Plant_GasTurbine_B.TmpSignalConversionAtCompressor[3] =
  Plant_GasTurbine_B.AmbientEnvtoEngine[2];
Plant_GasTurbine_B.TmpSignalConversionAtCompressor[4] =
  Plant_GasTurbine_B.AmbientEnvtoEngine[3];
Plant_GasTurbine_B.TmpSignalConversionAtCompressor[5] =
  Plant_GasTurbine_U.Plant_In.OuterLoopPlantFeedback.Nmech;
Plant_GasTurbine_B.TmpSignalConversionAtCompressor[6] =
  Plant_GasTurbine_U.IS_In[1];
Plant_GasTurbine_B.TmpSignalConversionAtCompressor[7] =
  Plant_GasTurbine_P.Constant4_Value;
Plant_GasTurbine_B.TmpSignalConversionAtCompressor[8] =
  Plant_GasTurbine_P.Compressor_s_C_Nc_M;
Plant_GasTurbine_B.TmpSignalConversionAtCompressor[9] =
  (Plant_GasTurbine_P.HP_Mods_Value[0] * rtb_s_C_Eff +
   Plant_GasTurbine_P.Constant_Value_a) *
  Plant_GasTurbine_P.Compressor_s_C_Wc_M;
Plant_GasTurbine_B.TmpSignalConversionAtCompressor[10] =
  (Plant_GasTurbine_P.HP_Mods_Value[1] * rtb_s_C_Eff +
   Plant_GasTurbine_P.Constant1_Value_i) *
  Plant_GasTurbine_P.Compressor_s_C_PR_M;
Plant_GasTurbine_B.TmpSignalConversionAtCompressor[11] =
  (Plant_GasTurbine_P.HP_Mods_Value[2] * rtb_s_C_Eff +
   Plant_GasTurbine_P.Constant2_Value) *
  Plant_GasTurbine_P.Compressor_s_C_Eff_M;
```

```
compressorStruct.NcDes = Plant_GasTurbine_P.Compressor_NcDes;
compressorStruct.PRDes = Plant_GasTurbine_P.Compressor_PRDes;
compressorStruct.EffDes = Plant_GasTurbine_P.Compressor_EffDes;
compressorStruct.RlineDes = Plant_GasTurbine_P.Compressor_RlineDes;
compressorStruct.IDes = Plant_GasTurbine_P.Compressor_iDesign_M;
compressorStruct.CustBldEn = Plant_GasTurbine_P.Compressor_CBLDEN_M;
compressorStruct.FBldEn = Plant_GasTurbine_P.Compressor_FBLDEN_M;
compressorStruct.CustBldNm = Plant_GasTurbine_P.Compressor_CustBldNm;
compressorStruct.FracBldNm = Plant_GasTurbine_P.Compressor_FracBldNm;
compressorStruct.Y_C_Map_NcVec = (double*)
  (Plant_GasTurbine_P.Compressor_Y_C_Map_NcVec_M);
compressorStruct.X_C_RlineVec = (double*)
  (Plant_GasTurbine_P.Compressor_X_C_RlineVec_M);
compressorStruct.Z_C_AlphaVec = (double*)
  (&Plant_GasTurbine_P.Compressor_Z_C_AlphaVec);
compressorStruct.T_C_Map_WcArray = (double*)
  (Plant_GasTurbine_P.Compressor_T_C_Map_WcArray_M);
compressorStruct.T_C_Map_PRArray = (double*)
  (Plant_GasTurbine_P.Compressor_T_C_Map_PRArray_M);
compressorStruct.T_C_Map_EffArray = (double*)
  (Plant_GasTurbine_P.Compressor_T_C_Map_EffArray_M);
compressorStruct.FracCusBldht = (double*)
  (&Plant_GasTurbine_P.Compressor_FracCusBldht);
compressorStruct.FracCusBldPt = (double*)
  (&Plant_GasTurbine_P.Compressor_FracCusBldPt);
compressorStruct.FracBldht = (double*)
  (&Plant_GasTurbine_P.Compressor_FracBldht);
compressorStruct.FracBldPt = (double*)
  (&Plant_GasTurbine_P.Compressor_FracBldPt);
compressorStruct.X_C_Map_WcSurgeVec = (double*)
  (Plant_GasTurbine_P.Compressor_X_C_Map_WcSurgeVec);
compressorStruct.T_C_Map_PRSurgeVec = (double*)
  (Plant_GasTurbine_P.Compressor_T_C_Map_PRSurgeVec);
compressorStruct.A = 14;
compressorStruct.B = 11;
compressorStruct.C = 1;
compressorStruct.D = 14;
compressorStruct.WcMapCol = Plant_GasTurbine_P.Compressor_WcMapCol;
compressorStruct.PRMapCol = Plant_GasTurbine_P.Compressor_PRMapCol;
compressorStruct.EffMapCol = Plant_GasTurbine_P.Compressor_EffMapCol;
compressorStruct.WcMapRw = Plant_GasTurbine_P.Compressor_WcMapRw;
compressorStruct.PRMapRw = Plant_GasTurbine_P.Compressor_PRMapRw;
compressorStruct.EffMapRw = Plant_GasTurbine_P.Compressor_EffMapRw;
compressorStruct.WcMapLay = Plant_GasTurbine_P.Compressor_WcMapLay;
compressorStruct.PRMapLay = Plant_GasTurbine_P.Compressor_PRMapLay;
compressorStruct.EffMapLay = Plant_GasTurbine_P.Compressor_EffMapLay;
compressorStruct.IWork = &Plant_GasTurbine_DW.Compressor_IWORK.Errors[0];
```

```
{
  SimStruct *rts = Plant_GasTurbine_M->childSfunctions[4];
  sfcnOutputs(rts,0);
}
```

```
Compressor_TMATS_body((real_T*)&Plant_GasTurbine_B.Compressor_o1[0], (real_T*)
                      &Plant_GasTurbine_B.CustomerBleedCharacteristics[0],
                      (real_T*)
                      &Plant_GasTurbine_B.FractionalBleedFlowCharacterist[0],
                      (real_T*)
                      &Plant_GasTurbine_B.TmpSignalConversionAtCompressor[0],
                      (real_T*)
                      &Plant_GasTurbine_P.CustomerorflowbasedBleedDemandV,
                      (real_T*)
                      &Plant_GasTurbine_P.FractionalBleedDemandVector_Val,
                      &compressorStruct);
```

```
/* Model initialize function */
void Plant_GasTurbine_initialize(void)
{
  /* Registration code */

  /* initialize non-finites */
  rt_InitInfAndNaN(sizeof(real_T));

  /* initialize real-time model */
  (void) memset((void *)Plant_GasTurbine_M, 0,
                sizeof(RT_MODEL_Plant_GasTurbine_T));
  rtsiSetSolverName(&Plant_GasTurbine_M->solverInfo,"FixedStepDiscrete");
  Plant_GasTurbine_M->solverInfoPtr = (&Plant_GasTurbine_M->solverInfo);

  /* Initialize timing info */
  {
    int_T *mdlTsMap = Plant_GasTurbine_M->Timing.sampleTimeTaskIDArray;
    mdlTsMap[0] = 0;
    Plant_GasTurbine_M->Timing.sampleTimeTaskIDPtr = (&mdlTsMap[0]);
    Plant_GasTurbine_M->Timing.sampleTimes =
      (&Plant_GasTurbine_M->Timing.sampleTimesArray[0]);
    Plant_GasTurbine_M->Timing.offsetTimes =
      (&Plant_GasTurbine_M->Timing.offsetTimesArray[0]);

    /* task periods */
    Plant_GasTurbine_M->Timing.sampleTimes[0] = (0.21);

    /* task offsets */
    Plant_GasTurbine_M->Timing.offsetTimes[0] = (0.0);
  }

  rtmSetTPtr(Plant_GasTurbine_M, &Plant_GasTurbine_M->Timing.tArray[0]);

  {
    int_T *mdlSampleHits = Plant_GasTurbine_M->Timing.sampleHitArray;
    mdlSampleHits[0] = 1;
    Plant_GasTurbine_M->Timing.sampleHits = (&mdlSampleHits[0]);
  }

  rtmSetTFinal(Plant_GasTurbine_M, 10.5);
  Plant_GasTurbine_M->Timing.stepSize0 = 0.21;

  /* Setup for data logging */
  {
    static RTWLogInfo rt_DataLoggingInfo;
    rt_DataLoggingInfo.loggingInterval = NULL;
    Plant_GasTurbine_M->rtwLogInfo = &rt_DataLoggingInfo;
  }

  /* Setup for data logging */
```

+1,400 lines of code

```
/* Model initialize function */
void Plant_GasTurbine_initialize(void)
{
  /* Registration code */

  /* initialize error status */
  rtmSetErrorStatus(Plant_GasTurbine_M, (NULL));

  /* block I/O */
  (void) memset(((void *) &Plant_GasTurbine_B), 0,
                sizeof(B_Plant_GasTurbine_T));

  /* states (dwork) */
  (void) memset((void *)&Plant_GasTurbine_DW, 0,
                sizeof(DW_Plant_GasTurbine_T));

  /* external inputs */
  (void)memset((void *)&Plant_GasTurbine_U, 0, sizeof(ExtU_Plant_GasTurbine_T));

  /* external outputs */
  (void) memset((void *)&Plant_GasTurbine_Y, 0,
                sizeof(ExtY_Plant_GasTurbine_T));
}
```

```
/* Model terminate function */
void Plant_GasTurbine_terminate(void)
{
  /* Terminate for S-Function (Ambient_TMATS): '<S2>/Ambient Env to Engine' */
  /* Level2 S-Function Block: '<S2>/Ambient Env to Engine' (Ambient_TMATS) */
  {
    SimStruct *rts = Plant_GasTurbine_M->childSfunctions[3];
    sfcnTerminate(rts);
  }

  /* Terminate for S-Function (Compressor_TMATS): '<S4>/Compressor' incorporates:
   *  Constant: '<S4>/Customer or flow based Bleed Demand Vector'
   *  Constant: '<S4>/Fractional Bleed Demand Vector'
   */
  /* Level2 S-Function Block: '<S4>/Compressor' (Compressor_TMATS) */
  {
    SimStruct *rts = Plant_GasTurbine_M->childSfunctions[4];
    sfcnTerminate(rts);
  }

  /* Terminate for S-Function (Burner_TMATS): '<S3>/Burner' */
  /* Level2 S-Function Block: '<S3>/Burner' (Burner_TMATS) */
  {
    SimStruct *rts = Plant_GasTurbine_M->childSfunctions[5];
    sfcnTerminate(rts);
  }

  /* Terminate for S-Function (Turbine_TMATS): '<S8>/Turbine' */
  /* Level2 S-Function Block: '<S8>/Turbine' (Turbine_TMATS) */
  {
    SimStruct *rts = Plant_GasTurbine_M->childSfunctions[6];
    sfcnTerminate(rts);
  }

  /* Terminate for S-Function (Nozzle_TMATS): '<S6>/Nozzle' */
  /* Level2 S-Function Block: '<S6>/Nozzle' (Nozzle_TMATS) */
  {
    SimStruct *rts = Plant_GasTurbine_M->childSfunctions[7];
    sfcnTerminate(rts);
  }

  /* Terminate for Enabled SubSystem: '<S4>/Scalar_To_Workspace' */

  /* Terminate for S-Function (DO_TMATS): '<S11>/DataOutput' */
  /* Level2 S-Function Block: '<S11>/DataOutput' (DO_TMATS) */
  {
    SimStruct *rts = Plant_GasTurbine_M->childSfunctions[0];
    sfcnTerminate(rts);
  }

  /* End of Terminate for SubSystem: '<S4>/Scalar_To_Workspace' */

  /* Terminate for Enabled SubSystem: '<S6>/C_To_Workspace' */

  /* Terminate for S-Function (DO_TMATS): '<S13>/DataOutput' */
  /* Level2 S-Function Block: '<S13>/DataOutput' (DO_TMATS) */
  {
    SimStruct *rts = Plant_GasTurbine_M->childSfunctions[1];
    sfcnTerminate(rts);
  }

  /* End of Terminate for SubSystem: '<S6>/C_To_Workspace' */

  /* Terminate for Enabled SubSystem: '<S8>/Scalar_To_Workspace' */

  /* Terminate for S-Function (DO_TMATS): '<S16>/DataOutput' */
  /* Level2 S-Function Block: '<S16>/DataOutput' (DO_TMATS) */
  {
    SimStruct *rts = Plant_GasTurbine_M->childSfunctions[2];
    sfcnTerminate(rts);
  }

  /* End of Terminate for SubSystem: '<S8>/Scalar_To_Workspace' */
}
```

MathWorks

```
/* Model terminate function */
void Plant_GasTurbine_terminate(void)
{
  /* (no terminate code required) */
}
```

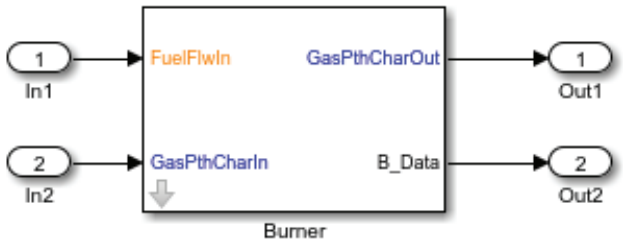
Total lines of code

Old version: 1,832
New version: 415

MathWorks

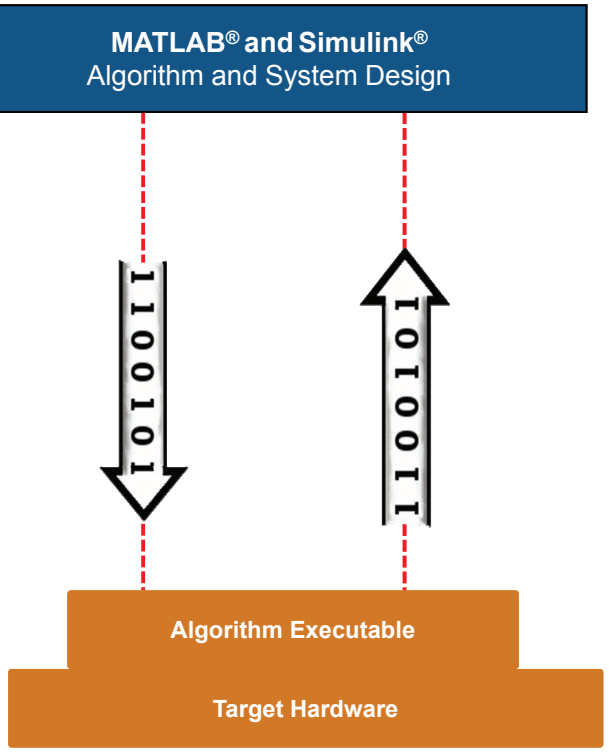
Test Beds

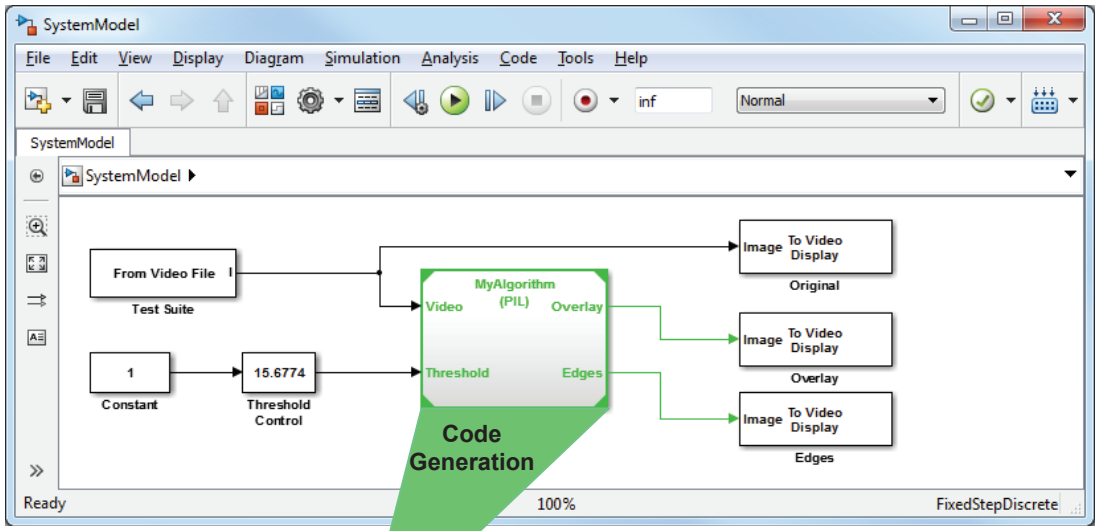
- Created test beds for each T-MATS block to ensure that functionality did not change.
- Compared old blocks against new blocks and generated code.
- Used input data from T-MATS example models and old NASA test models.



MathWorks

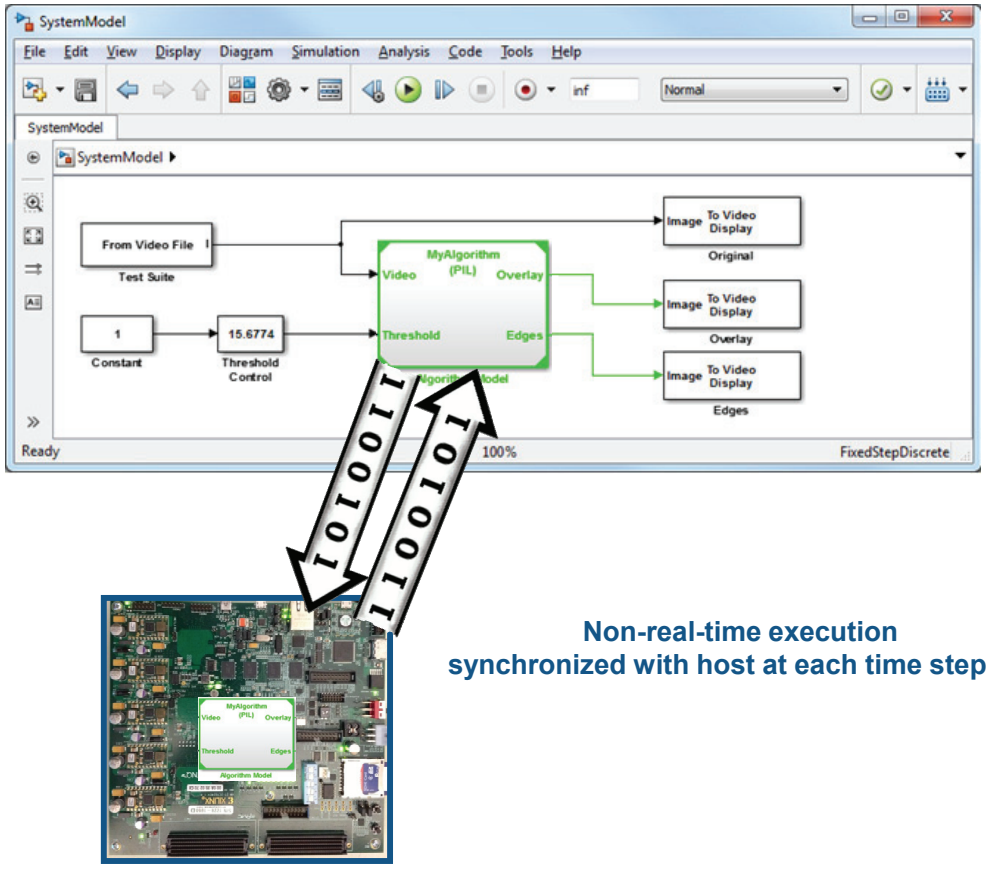
Processor-in-the-Loop (PIL) Simulation





Code Generation

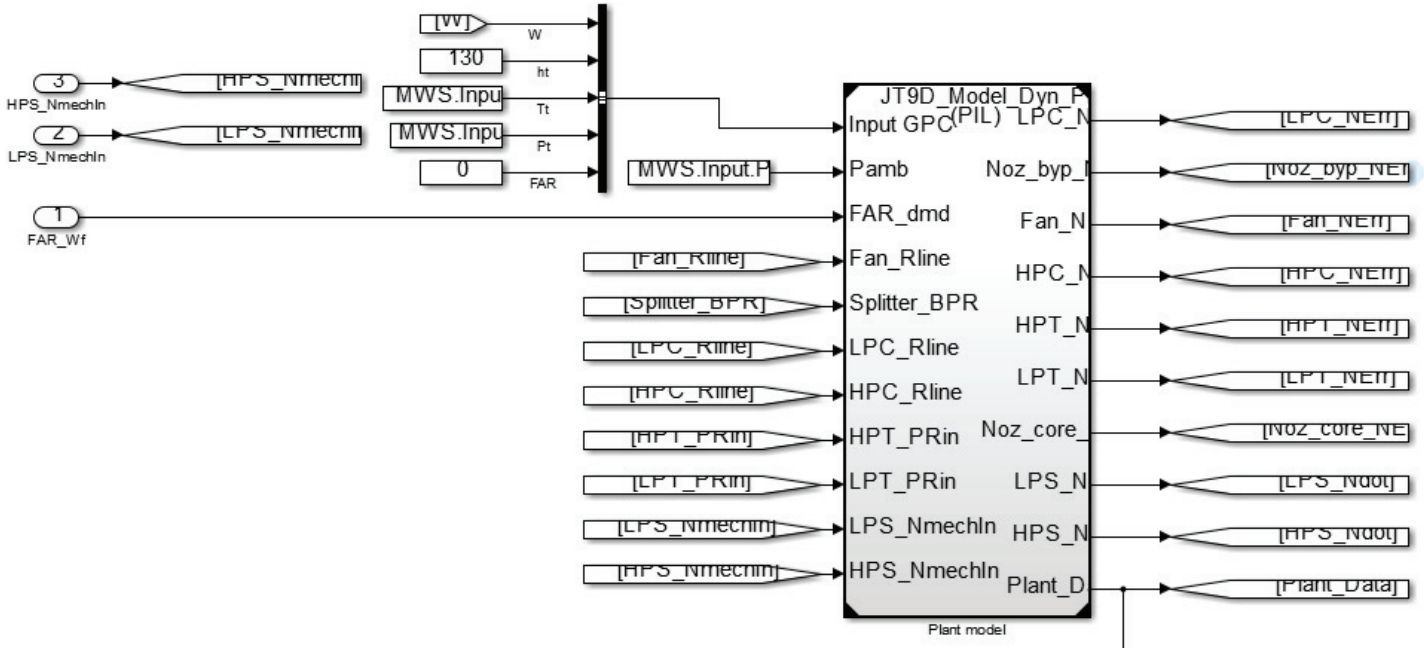
***BeagleBone Black* (PIL)**

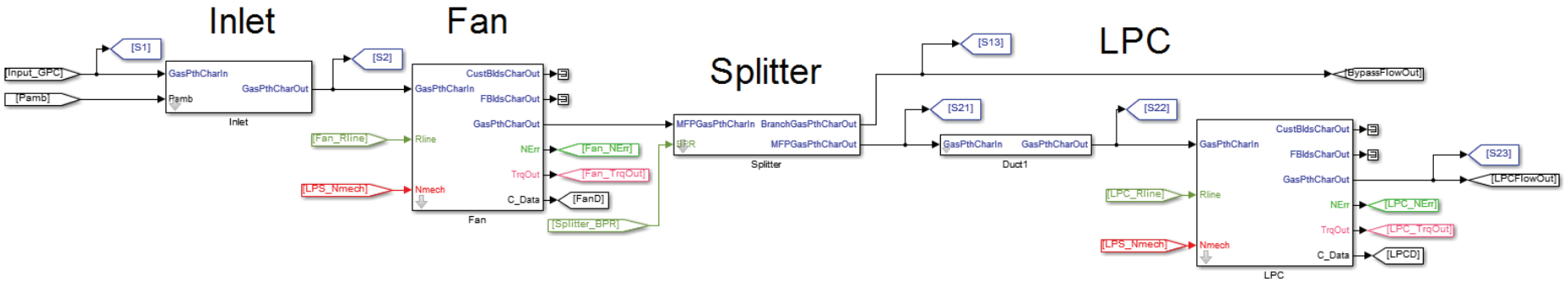


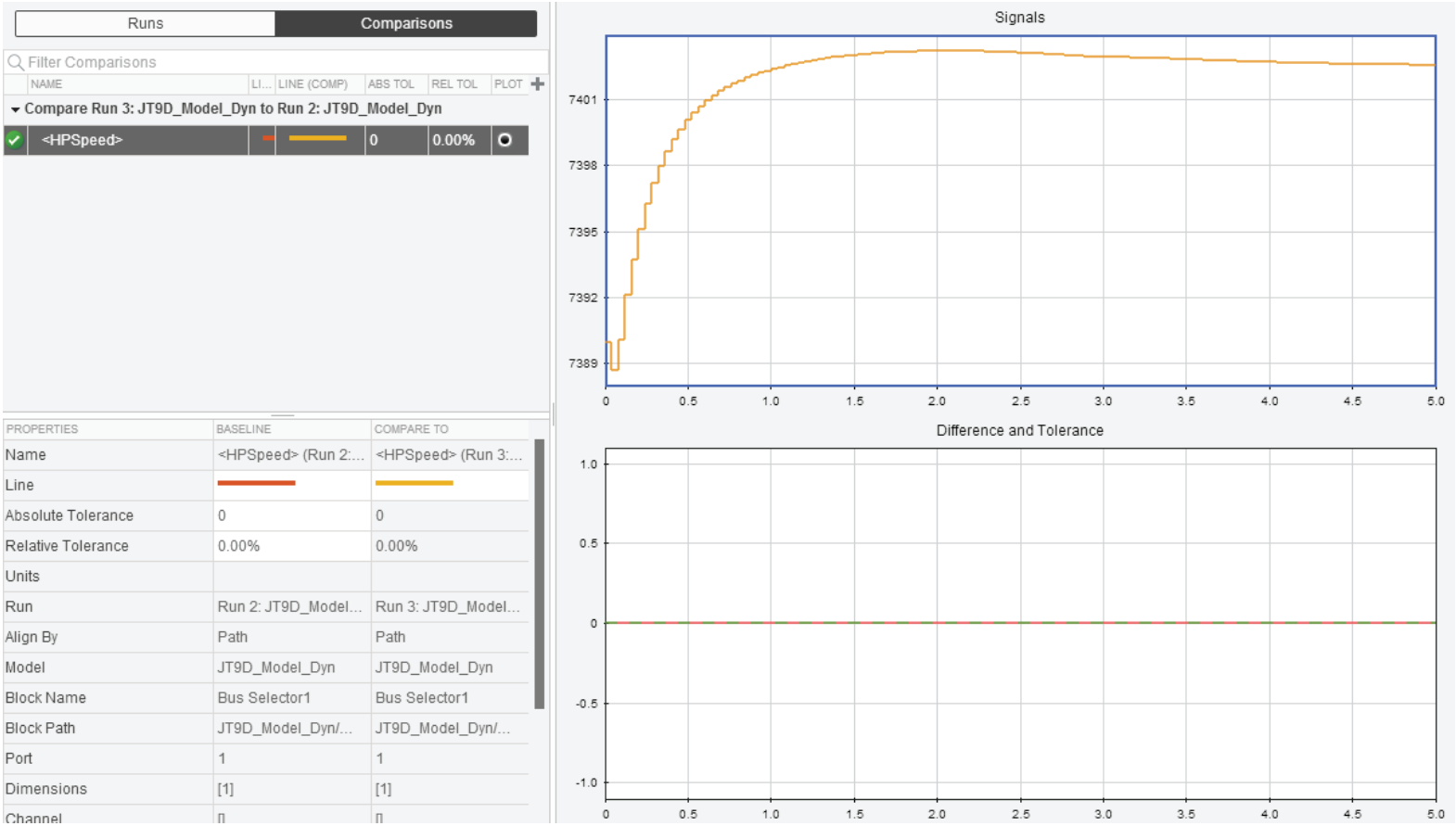
Non-real-time execution
synchronized with host at each time step

Demo

- Dynamic JT9D engine model running on BeagleBone Black hardware
- AM335X 1GHz ARM Cortex A8 processor


3
HPS_NmechIn
2
LPS_NmechIn
1
FAR_Wf
W
130
ht
Tt
Pt
0
FAR
MWS.Input.P
JT9D_Model_Dyn_P
(PIL)
Input GPC
Pamb
FAR_dmd
Fan_Rline
Splitter_BPR
LPC_Rline
HPC_Rline
HPT_PRin
LPT_PRin
LPS_NmechIn
HPS_NmechIn
LPC_N
Noz_byp_N
Fan_N
HPC_N
HPT_N
LPT_N
Noz_core_
LPS_N
HPS_N
Plant_D
Plant model


Inlet
Fan
Splitter
LPC
[S1]
[Input_GPC]
[Pamb]
GasPthCharIn
Pamb
GasPthCharOut
Inlet
[S2]
CustBldsCharOut
GasPthCharIn
FBldsCharOut
GasPthCharOut
[Fan_RLine]
Rline
NErr
[Fan_NErr]
TrqOut
[Fan_TrqOut]
[LPS_Nmech]
Nmech
C_Data
[FanD]
Fan
[Splitter_BPR]
[S13]
[BypassFlowOut]
MFPGasPthCharIn
BranchGasPthCharOut
BPR
MFPGasPthCharOut
Splitter
[S21]
GasPthCharIn
GasPthCharOut
Duct1
[S22]
[LPC_RLine]
[LPS_Nmech]
[S23]
[LPCFlowOut]
[LPC_NErr]
[LPC_TrqOut]
[LPCD]
LPC


Runs
Comparisons
Filter Comparisons
NAME
LI...
LINE (COMP)
ABS TOL
REL TOL
PLOT
Compare Run 3: JT9D_Model_Dyn to Run 2: JT9D_Model_Dyn
<HPSpeed>
0
0.00%
PROPERTIES
BASELINE
COMPARE TO
Name
<HPSpeed> (Run 2:...
<HPSpeed> (Run 3:...
Line
Absolute Tolerance
0
0
Relative Tolerance
0.00%
0.00%
Units
Run
Run 2: JT9D_Model...
Run 3: JT9D_Model...
Align By
Path
Path
Model
JT9D_Model_Dyn
JT9D_Model_Dyn
Block Name
Bus Selector1
Bus Selector1
Block Path
JT9D_Model_Dyn/...
JT9D_Model_Dyn/...
Port
1
1
Dimensions
[1]
[1]
Channel
[]
[]
Signals
7401
7398
7395
7392
7389
0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Difference and Tolerance
1.0
0.5
0
-0.5
-1.0
0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

JT9D Dynamic Engine Model on BeagleBone Black

Code Execution Profiling Report for JT9D_Model_Dyn_Plant

The code execution profiling report provides metrics based on data collected from a SIL or PIL execution. Execution times are calculated from data recorded by instrumentation probes added to the SIL or PIL test harness or inside the code generated for each component. See Code Execution Profiling for more information.

1. Summary

Total time (seconds × 1e-09)	104905090
Measured time display options	('Units', 'Seconds', 'ScaleFactor', '1e-09', 'NumericFormat', '%0.0f')
Timer frequency (ticks per second)	1e+09
Profiling data created	18-Aug-2017 16:52:36

2. Profiled Sections of Code

Section	Maximum Execution Time	Average Execution Time	Maximum Self Time	Average Self Time	Calls	
JT9D_Model_Dyn_Plant_initialize	65417	65417	65417	65417	1	
JT9D_Model_Dyn_Plant	1282708	595680	1282708	595680	176	

References

1. Chapman, J. W., Lavelle, T. M., May, R. D., Litt, J. S., Guo, T.-H., "Toolbox for the Modeling and Analysis of Thermodynamic Systems (T-MATS) User's Guide," NASA/TM—2014-216638, January 2014.
2. Chapman, J.W., Lavelle, T.M., May, R. D., Litt, J.S., and Guo, T. H., "Propulsion System Simulation Using the Toolbox for the Modeling and Analysis of Thermodynamic Systems (T-MATS)," NASA/TM—2014-218410, November 2014.
3. Zinnecker, A. M., Chapman, J.W., Lavelle, T.M., and Litt, J.S., "Development of a Twin-Spool Turbofan Engine Simulation Using the Toolbox for the Modeling and Analysis of Thermodynamic Systems (T-MATS)," NASA/TM-2014-218402, Nov. 2014.
4. Chapman, J. W., Lavelle, T. M., Litt, J. S., Guo, T.-H., "A Process for the Creation of T-MATS Propulsion System Models from NPSS data," AIAA 2014-3931, 50th AIAA/ASME/ SAE/ASEE Joint Propulsion Conference, July 2014, Cleveland, OH.
5. Lavelle, T. M., Chapman, J. W., May, R. D., Litt, J. S., Guo, T.-H., "Cantera Integration with the Modeling and Analysis of Thermodynamic Systems (T-MATS)," AIAA 2014-3932, 50th AIAA/ASME/SAE/ASEE Joint Propulsion Conference, Cleveland, OH, July 28-30, 2014.
6. Chapman, J.W., Lavelle, T.M., and Litt, J.S., "Practical Techniques for Modeling Gas Turbine Engine Performance," NASA/TM-2016-219147, July 2016.
7. Papathakis, K.V., Kloesel, K.J., Lin, Y., Clarke, S.C., Ediger, J.J., Ginn, S.R., "NASA Turbo-electric Distributed Propulsion Bench," AIAA 2016-4611, 52nd Joint Propulsion Conference, July 2016, Salt Lake City, UT.
8. Chapman, J.W., Guo, T.-H., Kratz, J.L., Litt, J.S., "Integrated Turbine Tip Clearance and Gas Turbine Engine Simulation," NASA/TM—2016-219146, October 2016.
9. Seok, J., Kolmanovsky, I., Girard, A., "Integrated/Coordinated Control of Aircraft Gas Turbine Engine and Power System: Towards Large Electrical Load Handling," 2016 IEEE 55th Conference on Decision and Control (CDC), December 12-14, 2016, Las Vegas, NV.
10. Aretskin-Hariton, E., Thomas, G., Kratz, J.L., Culley, D.E., "Design and Benchmarking of a Network-In-the-Loop Simulation for Use in a Hardware-In-the-Loop System," AIAA 2017-1943, 53rd Joint Propulsion Conference, Jan. 2017, Grapevine, Texas.
11. Connolly, J.W., Csank, J., Chicatelli, A., Franco, K., "Propulsion Controls Modeling for a Small Turbofan Engine," AIAA 2017-4787, 53rd Joint Propulsion Conference, Jan. 2017, Grapevine, Texas.
12. Dunham, W., Hencey, B., Kolmanovsky, I., Girard, A., "Predictive Propulsion and Power Control for Large Transient Power Loads in a More Electric Aircraft," 2017 American Control Conference, May 24–26, 2017, Seattle, WA.
13. Sousa, J., Paniagua, G., Collado Morata, E., "Thermodynamic analysis of a gas turbine engine with a rotating detonation combustor," Applied Energy, 195 (2017) 247–256.
14. Seok, J., Kolmanovsky, I., Girard, A., "Coordinated Model Predictive Control of Aircraft Gas Turbine Engine and Power System," Journal of Guidance, Control, and Dynamics, Vol. 40, No. 10 (2017), pp. 2538-2555.
15. Chapman, J.W., Litt, J.S., "Control Design for an Advanced Geared Turbofan Engine," AIAA-2017-4820, Propulsion and Energy Forum, June 10-12, Atlanta, GA.